**Faculdade de Engenharia da Universidade do Porto**

FEUP

# Green Computing

Sérgio Daniel Tristão Alves

Dissertação realizado no âmbito do
Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major Telecomunicações

Orientador: Tito Carlos Soares Vieira

Junho de 2010

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

**MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES** 2009/2010

A Dissertação intitulada

"GREEN COMPUTING"

foi aprovada em provas realizadas em 22/Julho/2010

**o júri**

Presidente Professora Doutora Ana Maria Rodrigues de Sousa Faria de Mendonça
Professora Associada do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Paulo Sérgio Tenreiro de Magalhães
Professor Auxiliar da Universidade Católica Portuguesa

Engenheiro Tito Carlos Soares Vieira
Professor Auxiliar Convidado do Departamento de Engenharia Informática da Faculdade de Engenharia da Universidade do Porto (Orientador).

O autor declara que a presente dissertação (ou relatório de projecto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extractos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são correctamente citados.

Autor - **Sérgio Daniel Tristão Alves**

Faculdade de Engenharia da Universidade do Porto

# Resumo

Este trabalho surge como parte de um projecto proposto pelo CICA (Centro de Informática Prof. Correia de Araújo) que tem como objectivo estudar o fenómeno de *Green Computing*. Este é um fenómeno crescente nos dias de hoje, em que os *datacenters* têm evoluído cada vez mais, sendo de maiores dimensões e com mais e melhores recursos, com a computação de elevado desempenho a ter uma performance cada vez mais optimizada. É apresentada uma visão global do estado da arte relativo a infra-estruturas de computação de elevado desempenho, como sistemas de *Grid Computing*, *Cloud Computing* e *Clusters* e tecnologias associadas, como *softwares* de escalonamento e monitorização. Estes são sistemas em que o *Green Computing* pode ser muito importante, no sentido de alcançar uma computação mais eficiente energeticamente e, por isso, mais ecológica e com menos impacto ambiental, sendo esse o grande objectivo do *Green Computing*. Com este objectivo alcançado, as organizações que usem infra-estruturas do tipo mencionado anteriormente podem também ver os seus custos operacionais reduzidos e, quando aplicável, ver os seus lucros melhorados. Também algumas soluções já actualmente implementadas são apresentadas, sendo que algumas delas podem ser objecto de posteriores melhorias ou tomadas como ponto de partida para construir outras soluções. Os parâmetros energéticos existentes que os *datacenters* devem seguir são também objecto de estudo, verificando se há normas que já são postas em prática hoje em dia ou que estão na calha para serem *standards* que regulam o funcionamento dos *datacenters*. Algumas organizações como a Green Grid têm contribuído bastante para o aparecimento destes parâmetros ou *standards.* Posteriormente, mostra-se também a metodologia idealizada para obter uma computação mais eficiente energeticamente, explicando o método idealizado para alcançar esse objectivo. Esta metodologia relaciona-se com o escalonador dos *Clusters* da FEUP, o *software* Moab, e tem em conta alguns dos parâmetros energéticos estudados. Esta metodologia tenta evitar que haja um número excessivo de máquinas ligadas nos *datacenters* sem estarem a fazer qualquer "trabalho útil", desperdiçando assim energia inutilmente. É descrito também o protótipo implementado com base na solução idealizada, sendo que os resultados dos testes efectuados sobre a estrutura GridFEUP são posteriormente expostos e discutidos.

# Abstract

*This report comes as part of a project proposed by CICA (Centro de Informática Prof. Correia de Araújo) which aims to study the phenomenon of Green Computing. This is a growing phenomenon today, where datacenters have increasingly evolved, being larger and with more and better resources, with high performance computing being more and more optimized. An overview of the state of the art regarding high performance computing infrastructures, like systems of Grid Computing, Cloud Computing and Clusters and associated technologies, such as software for scheduling and monitoring, is presented. These are systems where Green Computing can be very important in order to achieve a more energy efficience and therefore cleaner and with less environmental impact computing process, which is the main goal of Green Computing. By achieving this goal, organizations that use the previously mentioned kind of infrastructures can also see their lower operating costs and, when appliable, their profits improved. There are also some solutions currently implemented which are presented, some of which may be subsequently improved or taken as a starting point for building other solutions. The energy parameters which exist today that datacenters should follow are also the subject of study by checking whether there are standards that are already implemented today or in the pipeline to be standards governing the operation of datacenters. Some organizations like the Green Grid have greatly contributed to the emergence of these parameters or standards. Later, the idealized methodology for a more energy efficient computing process is showed, explaining the method idealized to achieve this objective. This methodology relates to the scheduler of FEUP's Clusters, Moab software and takes into account some of the energy parameters studied beforehand. This methodology tries to avoid having an excessive number of machines which are on in the datacenter and don't do any "useful work", thus wasting energy unnecessarily. Also described is an implemented prototype based on the idealized solution, and the results of the tests carried out on the structure GridFEUP are then shown and discussed.*

# Índice

# Lista de figuras

# Lista de tabelas

# Abreviaturas

Lista de abreviaturas (ordenadas por ordem alfabética)

| | |
|---|---|
| ACPI | *Advanced Configuration and Power Interface* |
| API | *Application Programming Interface* |
| ASHRAE | *American Society of Heating, Refrigerating and Air-Conditioning Engineers* |
| CICA | Centro de Informática Prof. Correia de Araújo |
| $CO_2$ | Dióxido de Carbono |
| CPE | *Compute Power Efficiency* |
| CPU | *Central Processing Unit* |
| CVS | *Coordinated Voltage Scaling* |
| CRAC | *Computer Room Air Conditioning* |
| CRAH | *Computer Room Air-Handling unit* |
| DCeP | *DataCenter energy Productivity* |
| DCiE | *DataCenter infrastructure Efficiency* |
| EARI | *Energy Aware Reservation Infrastructure* |
| EPA | *Environmental Protection Agency* |
| FEUP | Faculdade de Engenharia da Universidade do Porto |
| FSC | *Fan Speed Control* |

| | |
|---|---|
| GIPC | *Green IT Promotion Council* |
| GPU | *Discrete Graphics Processing Unit* |
| IaaS | *Infrastructure as a Service* |
| ÏANOS | *Intelligent ApplicationN-Oriented Scheduling* |
| IP | *Internet Protocol* |
| IVS | *Independent Voltage Scaling* |
| I&D | Investigação e Desenvolvimento |
| I/O | *Input/Output* |
| kW/h | kiloWatt por hora |
| LAN | *Local Area Network* |
| MAC | *Media Access Control* |
| MW | MegaWatts |
| OV | Organização Virtual |
| PaaS | *Platform as a Service* |
| PSU | *Power Supply Unit* |
| PUE | *Power Usage Effectiveness* |
| QoS | *Quality of Service* (Qualidade de Serviço) |
| SaaS | *Software as a Service* |
| SMP | *Symmetric MultiProcessing* |
| TCO | *Total Cost of Ownership* |
| TEC | *Typical Energy Consumption* |
| TI | Tecnologias de Informação |
| TGV | *Train* à *Grande Vitesse* |
| UPS | *Uninterrupted Power Supply* |
| VOVO | *Vary-On Vary-Off* |
| W | Watt |
| W/h | Watt por hora |

# Capítulo 1

# Introdução

Neste capítulo são apresentadas considerações sobre o enquadramento do projecto a efectuar, dando a conhecer o CICA e a sua estrutura. De seguida, far-se-á uma descrição geral da problemática que dá origem ao *Green Computing*. Ainda serão expostos os objectivos e a motivação do projecto, finalizando com uma vista sobre a organização da dissertação.

## 1.1 - Enquadramento do Trabalho

O tema da dissertação é *Green Computing*, sendo que esta está inserida num projecto do Centro de Informática Prof. Correia de Araújo (CICA) proposto pelo Professor Tito Carlos Soares Vieira, que será também orientador da dissertação e que acumula, entre outros, o cargo de Director de Serviços do CICA.

Nos últimos anos, observou-se um aumento do número de aplicações ou projectos que necessitam de um desempenho computacional elevado, seja pela necessidade de analisar grande quantidade de dados, como de armazenar enorme quantidade de informação e partilhá-la oferecendo qualidade de serviço e rapidez na execução de aplicações. Este aumento originou um grande e crescente interesse em sistemas como o *Grid Computing*, *Cloud Computing*, *Clusters*, etc. Estes são sistemas que fazem uso de uma cooperação entre múltiplos recursos de uma ou mais organizações, interligados directa ou indirectamente, heterogéneos ou homogéneos, permitindo deste modo, por exemplo, aumentar o poder computacional e capacidade de armazenamento, oferecer serviços ou mesmo expandir a conectividade de alta velocidade entre nós da rede, explorando as potencialidades dos recursos usados.

Neste contexto, o problema que é objecto de estudo e investigação neste trabalho é de vital importância para empresas ou entidades que sejam proprietárias ou administradoras de estruturas deste tipo. O CICA apresenta-se como um destes casos pois tem sob sua administração um sistema de *Clusters* de alta performance, por exemplo para apoio ao ensino pós-graduado e às actividades de Investigação e Desenvolvimento (I&D) da FEUP.

O CICA é um Serviço Central da Faculdade de Engenharia da Universidade do Porto (FEUP). A sua missão é disponibilizar e assegurar a operacionalidade de recursos e serviços

de informática para toda a comunidade da FEUP, promovendo a sua utilização e inovação.

A primeira prioridade do CICA é garantir que os utilizadores dos sistemas geridos por esta instituição contem com uma disponibilidade e operacionalidade elevada desses sistemas, como é o caso, entre outros, dos sistemas de correio electrónico, impressão e informação da FEUP e, claro, o sistema de *Clusters* de alta performance já referido. É outro dos objectivos do CICA assegurar que os serviços que fornece tenham condições adequadas de desempenho, segurança e capacidade e que estejam sempre actualizados de acordo com as últimas novidades tecnológicas ou a adequação às necessidades da comunidade FEUP.

Em segundo lugar, tem como prioridade a publicitação aos seus utilizadores dos recursos e serviços computacionais disponíveis, tal como dar apoio técnico na resolução de variados problemas que possam surgir na utilização desses recursos. É garantido pelo CICA que esse apoio seja o mais eficiente possível, para que a utilização dos recursos e serviços disponibilizados pelo CICA seja fomentado junto da comunidade FEUP. O CICA também procura criar no seu seio um ambiente de trabalho de qualidade para o ensino, investigação e administração da FEUP. A inovação, promovendo e estimulando a actualização e avanço tecnológico, é outra meta do CICA, que tenta sempre desenvolver, testar e disponibilizar novos serviços e recursos à sua comunidade de utilizadores.

A organização desta instituição é feita em quatro Unidades Funcionais [3], respeitando esta divisão às principais áreas de actividade do Centro: Administração de Sistemas, Infra-Estruturas e Redes de Comunicação, Sistemas de Informação e Microinformática e Suporte ao Utilizador.

A FEUP/CICA é uma das instituições de âmbito educacional e de investigação pioneiras a nível nacional em ter este género de infra-estruturas, administrando então três *Clusters* de alta performance. Um deles, o *Cluster* IBM eServer 1350, resulta de um projecto conjunto entre IBM e FEUP com vista a aumentar o poder de cálculo na investigação em diversas áreas como o caso das engenharias química, mecânica, electrónica e tecnologias de informação. Este *Cluster* possui 32 nós, cada nó com 2 processadores, com cerca de 132 GB de memória RAM e 3,45 TB (raw) de espaço em disco. Os outros dois *Clusters* geridos pelo CICA são o Idmec e o Ineb. O Idmec divide-se em Idmeca e Idmecb, tendo no total 39 nós. Cada um destes nós tem dois *cores*, no entanto, devido a um processo de *hyperthread*, a sua capacidade aumenta para 4 processadores operacionais, o que faz com que este *Cluster* opere com 156 processadores e que cada nó tenha sensivelmente 8GB de memória RAM. O *Cluster* Ineb possui 18 nós, cada um com 4 processadores operacionais devido ao mesmo processo de *hyperthread* do *Cluster* Idmec. Cada nó do Ineb tem sensivelmente 4GB de memória RAM. Para gerir estes três *Clusters* o CICA associou-se a vários grupos de investigação e departamentos da FEUP e criou um consórcio, já que a utilização de recursos computacionais de grande dimensão não é muito viável se tiver de ser assegurada por uma só entidade de forma individual.

| Membros do Consórcio | Percentagem (%) |
|---|---|
| Direcção da FEUP/CICA | 10.5 |
| LSRE | 39.2 |
| CESA | 22.4 |
| IDMEC | 2.8 |
| CEHRA | 2.8 |
| DEC | 2.8 |
| DEEC | 2.8 |
| MEI | 2.8 |
| MEEC | 2.8 |
| MTM | 2.8 |
| MMCCE | 8.4 |

**Figura 1.1 –** Membros do Consórcio gestor da GridFEUP [3]

Assim foi criada a estrutura GridFEUP, que assegura a ligação entre os 3 sistemas e permite uma gestão mais centralizada destes. Como a GridFEUP possui recursos heterogéneos no seu todo, ligando vários *Clusters* de alta performance, torna-se necessário ter em atenção o consumo energético em estruturas de tão larga escala e que no presente já atingem tamanhos impressionantes. É, assim, cada vez mais evidente a preocupação em torno dos gastos de energia e, como consequência, gastos monetários destes sistemas, já que o consumo de potência é realmente uma porção bastante significativa do preço de possuir uma estrutura deste género.

## 1.2 - O Problema

Será feita agora uma breve apresentação geral do problema em mãos que é tema desta dissertação.

O *Green Computing* incide sobre a problemática da eficiência energética associada, em particular, a estruturas de computação de elevado desempenho, como *Grids* computacionais, *Clusters* ou infra-estruturas que fazem uso de sistemas de *Cloud Computing*, por exemplo. Hoje em dia nota-se uma tendência mundial para uma preocupação cada vez maior com o estado do planeta e do ambiente. Este facto acontece muito por culpa das alterações climáticas e comportamentos crescentemente alterados da natureza, tanto ao nível da frequência de desastres naturais como furacões, sismos ou outros como vagas de frio ou de calor extremos, só para referir alguns. Outro ponto importante para a grande preocupação ambiental de hoje é o facto de algumas fontes de energia começarem a escassear, como o petróleo, e o grande desperdício de energia eléctrica desnecessariamente o que contribui para o grande mal identificado presentemente à escala mundial, o aquecimento global. Várias campanhas têm sido feitas no sentido de alertar o público para esta realidade, promovendo

acções tão simples como desligar os aparelhos domésticos quando não estão em uso ou luzes de divisões que não se encontrem ocupadas. No entanto, estes são casos singulares, sem grande impacto no problema e embora podendo dar ajuda na resolução deste, a maior fatia contribuinte para este problema reside nas grandes corporações que produzem, utilizam e desperdiçam grande quantidade de energia. Neste caso incluem-se as infra-estruturas de elevado desempenho computacional referidas e que também objecto de estudo desta dissertação.

Em sistemas distribuídos móveis ou que tenham a restrição de uso de baterias, a poupança de energia tem sido matéria de estudo há já algum tempo. No entanto, para sistemas distribuídos não móveis e de larga escala, como *Clusters* ou *Grids* computacionais, este tópico apenas recentemente tem vindo a ser tomado em consideração. Isto resulta da circunstância do consumo de energia nestas infra-estruturas vir a aumentar a um ritmo elevadíssimo, já que também estes sistemas vão crescendo, seja devido à necessidade de aumentar o seu desempenho para melhor performance nas suas funções, seja por ter de elevar o número de recursos para suprir a maior procura e utilização destas estruturas por um número cada vez maior de pessoas ou organizações. Estas infra-estruturas de computação de elevado desempenho têm o intuito de resolver problemas que requerem muitos recursos em termos de equipamento e comunicação. No entanto, é sabido que por exemplo os *Clusters* ou *Grids* não são sempre usados na sua capacidade total, sendo que num trabalho anterior sobre *Grids* operacionais é mostrado até que são usados apenas entre 60% a 80% [1]. Como explicitado noutro trabalho [2] sobre o tema, numa análise feita numa *Grid* operacional de Lyon, França, durante o ano de 2007, a estrutura referida consumiu 172500 kWh nesse período. Estes valores não incluem os equipamentos de rede ou o sistema de refrigeração, apenas os nós. Este consumo é igual ao de um TGV que ande mais que 11650 km, sendo que o consumo de potência de um TGV é cerca de 14.79 kWh por km.

O impacto ambiental destes *datacenters* (como são os *Clusters*, *Grids* ou sistemas de *Cloud Computing*) é deste modo enorme, sendo que estudos recentes descobriram que as emissões de dióxido de carbono ($CO_2$) de vários *datacenters* ultrapassam as emissões de muitos países. A isto ainda se junta o facto de muitos dos componentes do equipamento computacional conter substâncias tóxicas como mercúrio ou chumbo, que provoca danos ambientais, por exemplo, aquando do despejo de material já ultrapassado em lixeiras.

Portanto, é possível aferir que o consumo de potência nestes *datacenters* distribuídos e de larga escala é enorme. Obviamente que com um grande consumo de energia vem também associado outro problema de enorme importância para as organizações que fazem uso destas estruturas: o preço a pagar pelo uso de tão elevada quantidade de energia, sendo que não só a energia consumida em si será cara, como todos os sistemas de refrigeração e comunicação associados a estes sistemas de larga escala têm os seus custos. As contas associadas a este consumo desmedido começam então a representar uma grande percentagem das despesas das entidades e organizações que fazem uso destes sistemas.

Assim, as grandes despesas inerentes à baixa eficiência energética e a grande pegada ambiental provocada pelo seu desempenho levam a que governos e os membros envolvidos neste campo como investigadores e consumidores estejam a focar atenções no desenvolvimento de novas tecnologias e estratégias que permitam atingir uma computação mais verde. Não entrar neste processo de tornar mais eficiente e mais amiga do ambiente a computação nestes *datacenters* seria avassalador tanto para o ambiente como para os orçamentos das instituições envolvidas neste processo.

## 1.3 - Objectivos

Os objectivos deste trabalho são, em primeira instância, analisar os desenvolvimentos em infra-estruturas de computação de elevado desempenho, como sistemas de *Cloud Computing*, *Clusters* ou *Grids* operacionais, tal como identificar algumas tecnologias ou métodos existentes para atingir uma maior eficiência energética nestes *datacenters*. Outro dos objectivos é desenvolver uma metodologia que vise reduzir os consumos energéticos e, portanto, aumentar a eficiência energética. Esta metodologia também poderá envolver parâmetros de eficiência energética relevantes para os *datacenters*. O estudo e investigação destes parâmetros de eficiência energética apropriados para estruturas deste género é também um objectivo desta dissertação. Por fim, está prevista a implementação de um protótipo sobre a infra-estrutura GridFEUP com base na metodologia definida.

## 1.4 - Justificação e Motivação

Nos dias de hoje, nota-se, uma preocupação crescente ao nível da eficiência energética, principalmente na comunidade das TI.

Como visto na secção 1.2, as estruturas de computação de elevado desempenho têm uma pegada ecológica muito elevada. No entanto, estas estruturas raramente precisam de manter, por períodos alargados, picos de performance constantes. Assim, com o aumento do poder computacional destas estruturas, o consumo energético e os seus gastos monetários inerentes começam a ser uma dor de cabeça para os proprietários de *datacenters*. O próprio CICA, que tem debaixo da sua gestão não só os três *Clusters* de elevado desempenho como também um centro computacional com mais de 1000 computadores, tem tentado encontrar soluções para uma computação mais eficiente energeticamente. Por exemplo, ao constatar que a utilização das infra-estruturas das salas de informática não era regular ao longo do tempo foi desenvolvido em 2004 um *software* de monitorização para identificar eventuais padrões. Isto permitiu que se fosse reduzindo o número de computadores ligados mas inactivos, principalmente no período nocturno [3].

Por tudo que já foi dito sobre esta problemática, pode-se constatar que esta é bastante actual e pertinente e que começa a ser olhada agora como vital para a implementação de mais e maiores estruturas de elevado desempenho computacional com maior eficiência energética, sendo que o CICA não foge a esta regra.

Há assim vários factores que são a grande motivação da maior parte das organizações e entidades no caminho rumo a uma computação de elevado desempenho mais eficiente e optimizada, mais "verde" e amiga do ambiente e, claro, como consequência, mais barata e proveitosa.

Um desses factores é a restrição de capacidade dos centros de dados. Isto pode ser

explicado com base em várias razões específicas para estas restrições, como falta de espaço físico, a necessidade de expandir as infra-estruturas de geração de potência e refrigeração, falta de capacidade disponível do sistema de *Clusters*, *Grid*, etc e até questões devidas à densidade de potência localizadas e problemas de refrigeração em certas zonas específicas do sistema. Tudo isto representa restrições à capacidade da infra-estrutura usada.

Outro factor de motivação importante é a economia de custos, pois o consumo de energia representa uma fatia cada vez maior do orçamento de um centro de dados deste tipo.

Outra circunstância motivadora para a optimização destas infra-estruturas será a preocupação ambiental. Como se sabe, o impacto no ambiente destes centros de dados distribuídos de larga escala é tremendo, sendo em vários casos mais significativo que o impacto de certos países individuais. Um compromisso para atingir níveis de emissões e consumos energéticos aceitáveis e o menor possível é também um objectivo com força crescente no mundo empresarial de hoje.

Outros factores podem incluir preocupações sobre regulamentos e normas que possam estar na iminência de surgir e que limitem a operação do centro de dados, tentando assim antecipar estas legislações e directrizes reguladoras.

## 1.5 - Organização da Dissertação

Este trabalho está organizado da seguinte forma. No próximo capítulo será apresentado o estudo efectuado sobre o estado da arte referente ao tema da dissertação, o *Green Computing*, e sobre áreas tecnológicas inerentes a este tema e ferramentas relevantes. Portanto, apresentar-se-á uma visão do que é o *Green Computing* e qual a sua evolução no mundo actual, tanto a nível de entendimento e reconhecimento geral como a nível de soluções que vêm surgindo tendo por base uma computação mais “verde”. Também se apresentará o estudo sobre as áreas tecnológicas em que o *Green Computing* faz mais sentido, como o são as áreas de *Grid Computing*, *Clusters* ou sistemas de *Cloud Computing*. No capítulo 3 será apresentado o resultado da investigação realizada sobre os diferentes parâmetros e normas energéticas apropriados para infra-estruturas de computação de elevado desempenho como os *datacenters*. O capítulo 4 compreende a descrição da solução idealizada e do protótipo implementado. No capítulo seguinte serão apresentados e discutidos os resultados dos testes efectuados com base no protótipo implementado sobre a infra-estrutura GridFEUP. Finalmente, o capítulo 6 apresentará as conclusões do trabalho e referirá alguns trabalhos futuros possíveis tendo em conta a solução idealizada de forma a optimizar a mesma.

# Capítulo 2

# Estado da Arte

## 2.1 - Introdução ao estado da arte

Neste capítulo é apresentado o estudo do estado da arte. Este estudo baseia-se, numa primeira fase, na análise sobre *Green Computing*, o tema exacto da dissertação a realizar, nomeadamente uma visão geral sobre o tema e conceito do *Green Computing*, análise de questões estratégicas no âmbito do *Green Computing*, identificação dos seus obstáculos e análise de algumas soluções e plataformas existentes presentemente para atingir uma computação mais eficiente energeticamente. Depois será feita uma visão geral sobre o funcionamento geral, arquitecturas ou implementações físicas mais típicas e algumas iniciativas e projectos actuais sobre áreas tecnológicas de computação de elevado desempenho em que o conceito de *Green Computing* é cada vez mais importante, como são as áreas do *Grid Computing, Cloud Computing* e *Clusters*. Por fim, numa terceira fase, será realizada uma observação global sobre ferramentas relevantes para o projecto a realizar e cujo funcionamento geral deve estar presente. São os casos dos escalonadores e monitorizadores usados concomitantemente em sistemas deste tipo, oferecendo então uma visão geral de ferramentas específicas usadas na FEUP como o *Moab* (escalonador) e o *Ganglia* (monitorizador).

## 2.2 - Green Computing

### 2.2.1 – Visão Geral

O tópico da economia de energia tem-se tornado muito importante um pouco por todo o Mundo nos últimos anos. O domínio dos *datacenters* de larga escala, como o caso de infra-estruturas de computação de elevado desempenho distribuídas baseadas em *Grids* operacionais ou sistemas de *Clusters* ou *Cloud Computing*, não podia escapar a esta corrente, embora só recentemente este tópico tenha sido posto em cima da mesa de preocupações.

É certo que as chamadas Tecnologias de Informação e Comunicação (TIC), como as usadas nas infra-estruturas atrás indicadas, são importantes no desenvolvimento de métodos para aumentar a eficiência destes sistemas e reduzir os seus custos. As TIC são muitas vezes usadas para modelar, simular e optimizar estes processos. No entanto, as TIC e os seus centros de operação, como os sistemas mencionados, são responsáveis por um consumo elevadíssimo de electricidade e energia, sendo que já se começou a desenvolver trabalho com vista à redução dos custos económicos e ecológicos associados a esta realidade. Como referido numa publicação da especialidade [4], é estimado que os computadores no Mundo (cerca de 1.5 biliões) consumam cerca de 90000 MW de potência eléctrica, que é cerca de 10% do consumo de energia à escala global. Adicionalmente, o consumo de energia dos servidores mundiais duplicou entre 2000 e 2005. De acordo com estimativas recentes do fórum da CeBit em Hannover em 2008, o uso da Internet a nível mundial, com todos os computadores, servidores e tecnologias de comunicação associadas, produz o mesmo nível de dióxido de carbono que toda a indústria de aviação e precisa de um equivalente a catorze estações de potência.

Estas são razões mais que suficientes para justificar uma intervenção rápida e profunda sobre estes gastos energéticos. Tradicionalmente sempre houve pouca sensibilização para preocupações ecológicas no mundo da computação. Os impactos ecológicos deste processo em *datacenters* de larga escala sempre foram deixados para segundo plano, com o objectivo de maior performance, capacidade e capabilidades como grandes prerrogativas. No entanto, novas regulamentações e estudos governamentais já começam a ter em conta procedimentos de eficiência energética para servidores e *datacenters*. Existem já vários artigos que especificam melhores práticas para uma melhor rentabilização de infra-estruturas TIC e questões estratégicas sobre a gestão destes centros. Também os benefícios para a economia e imagem das organizações que têm preocupações ecológicas estão a ser tomadas em consideração.

Estas circunstâncias do mundo actual levam então a que se comece a olhar para o tema de *Green Computing* como o futuro da computação e das infra-estruturas TIC de larga escala, pois só assim a sua eficácia e o seu crescimento serão viáveis. Há já algumas iniciativas e organizações cujos trabalhos despoletaram grande interesse neste "despertar" ecológico, como nos Estados Unidos pelo movimento *Green Destiny* ou no Japão. Outras iniciativas surgiram para se dedicarem a esta problemática, como a *Green Internet* ou *Green500* [5], ou mesmo a organização sem fins lucrativos *The Green Grid* [6], que promove mesmo a adopção universal de standards, processos, métricas e tecnologia para melhorar a performance e eficiência energética em *datacenters*. Esta organização envolve mesmo grandes companhias

como são os casos da *Microsoft*, *Dell*, *Intel*, *Sun Microsystems*, *IBM* ou *VMWare*, para nomear algumas. A nível Europeu a importância dada a este tema ainda não é tão grande, mas já há também vários projectos dedicados a ele. São os casos dos projectos *AIM* e *BE-AWARE*, que estão a investigar abordagens complementares com diferentes perspectivas com vista à consciencialização ambiental das estruturas TIC. Também os projectos *EcoGrappe* e *GreenNet* em França estão a investigar como desenvolver uma computação mais ecológica e que economize energia. Mais recentemente, a *European COST Action IC0804* [7] tem tentado encorajar os investigadores deste campo a unir-se e cooperar de forma a chegar a novos desenvolvimentos neste tópico.

Este tipo de projectos e organizações é bastante importante já que é reconhecido que um esforço único e isolado não será suficientemente eficaz para provocar alterações de fundo na pegada ecológica das infra-estruturas TIC, já que estes *datacenters* têm um impacto ambiental muito extenso e diverso. É necessário assim um trabalho conjunto à escala mundial neste sentido, com uma abordagem multidisciplinar, juntando pesquisas e métodos de vários campos e empresas de forma a optimizar este processo de economia de energia.

Como se mencionado anteriormente neste trabalho, a indústria das TIC é olhada por muitos sectores da sociedade como uma das causas das alterações climáticas verificadas no presente. Por isso mesmo, é criticada a sua posição complacente em relação ao seu enorme impacto ambiental. A emissão de gases que conduzem ao efeito de estufa do planeta é estimada em cerca de 2%, similar à contribuição da indústria da aviação, e será cada vez maior à medida que aumenta a procura e exigência de mais poder computacional e performance e a dependência das indústrias em infra-estruturas de computação de grande desempenho e de larga escala. A preocupação até agora tem sido mais de ir de encontro a estas exigências, deixando para plano de fundo as preocupações da pegada ambiental que isso provoca. Vários planos estratégicos têm sido desenvolvidos e postos em prática no sentido de consciencializar as organizações para estes problemas e com o intuito de se atingir infra-estruturas com optimização de consumos de energia.

### 2.2.2 – Questões Estratégicas

Há quem defenda que este começa por ser um problema de fundo, ou seja, que começa no modo como são educados os tecnólogos que irão trabalhar no mundo das TIC [8]. Os defensores desta ideia baseiam-se no facto de os cursos, até muito recentemente ou ainda hoje, são direccionados somente para as capacidades técnicas e de resolver problemas e não terem uma componente de responsabilidade social. Assim, é patrocinada a estratégia de que os cursos que formam os profissionais de TIC devem consciencializar e mostrar as implicações ambientais da sua profissão no presente e no futuro. Também devem promover a discussão sobre que maneiras existem de resolver esta problemática, já que parece haver a ideia que só se atinge mais eficiência energética por substituição de *hardware* por outro novo mais eficaz e económico energeticamente, quando não é de todo esse o caso. Outra medida é a promoção de que a prática de uma computação mais "verde" será boa em termos de imagem, fazendo ver que estas preocupações ecológicas podem tornar os profissionais recém formados, neste

caso, mais atractivos na entrada do mundo do trabalho ou as empresas mais atractivas a vários clientes.

Outra estratégia que começa a ser empregue é o uso de ferramentas que ajudam a escolher soluções com vista à economia de energia em infra-estruturas de computação de elevado desempenho de larga escala. Como estas estruturas, *Grids* operacionais ou *Clusters*, variam muito na sua extensão, composição, design e requisitos, não há uma solução perfeita e única que se adapte a todo e qualquer sistema. Um exemplo de uma ferramenta deste tipo é a *Going Green Impact Tool* [9], que foi desenvolvida com base na necessidade de muitas empresas e organizações precisarem de uma análise extensiva e cuidada sobre o seu sistema de forma a escolher o melhor equilíbrio de soluções para um retorno óptimo do seu investimento. Esta ferramenta é importante sobretudo devido à necessidade de uma visão holística por parte das empresas, não só focando as questões relacionadas com eficiência energética mas também avaliar a situação de um ponto de vista económico e a longo prazo. Esta ferramenta permite então essa visão extensiva, ao mesmo tempo que suporta o processo de encontrar as melhores soluções para uma computação mais "verde" para cada caso e de as priorizar correctamente. São tidas em conta questões relacionadas com a energia em *datacenters*, tal como o impacto das soluções encontradas na eficiência energética e quantifica-se o Custo Total de Propriedade (TCO - *Total Cost of Ownership*), relacionando-as com cada negócio associado. As soluções para *Green Computing* correntemente incluídas nos cálculos desta ferramenta são a virtualização, a substituição de equipamento de TIC, desmantelamento de equipamento com baixa produtividade, gestão de potência, arrefecimento livre ou por líquido directo e reutilização de calor residual. Muitas destas soluções serão analisadas detalhadamente mais à frente nesta secção.

Focando agora atenção para estratégias em estruturas de menor escala, tem-se por exemplo as medidas implementadas pela *Alcatel-Lucent* [10], uma corporação líder no sector das telecomunicações. Uma dessas medidas foi a instalação de uma característica nova em praticamente todas as suas estações base sem fios implementadas desde 1999 que reduz o consumo de potência quando o tráfego baixa, com uma reactividade em tempo real que não afecta a qualidade de serviço.

A mesma corporação também implementou um programa chamado *Alternative Energy Programme*, com já cerca de 300 estações sem fios a serem alimentadas por energia solar. O objectivo é desenvolver as primeiras estações base sem fios alimentadas por energias alternativas, tornando possível a vários operadores estender o alcance dos seus serviços sem fios para pessoas que vivam em zonas não servidas por uma rede eléctrica por exemplo. Não só a energia solar é usada, como está também prevista a integração da energia eólica ou biocombustíveis.

Há ainda estratégias baseadas numa abordagem mais orientada ao nível do utilizador. Estas abordagens tentam sensibilizar os consumidores comuns de energia e alertá-los para o impacto desse consumo no ambiente de forma a provocar mudanças no seu comportamento. Isto é feito através de diversos projectos na área da energia e do design, ou seja, da forma como essa energia é mostrada aos consumidores. Actualmente a grande tendência é esconder ao máximo as fontes de energia, como contadores eléctricos, fios, tomadas ou baterias, seja em casas, prédios ou produtos. Por conseguinte, os utilizadores não ficam muito cientes dos gastos energéticos que sucedem. Portanto, um design que torne o uso de energia visível de uma forma atractiva pode ser uma ajuda importante para consciencializar os utilizadores e fazê-los reflectir sobre os consumos de energia que estão a realizar. O *Interactive Institute* da

Suécia assenta as suas iniciativas nas premissas descritas [11]. Como exemplo de designs energéticos deste tipo tem-se o *Power-Aware Cord* que é um fio de energia eléctrica em que o cordão está projectado para que quanto mais energia por ele passa, mais ele brilha. É ao mesmo tempo visualmente atractivo e uma lembrança da energia que está a ser dispendida. Outro exemplo é o *Energy-Aware Clock*, que por exemplo colocado numa cozinha dá informação sobre a carga de electricidade da casa, sendo que o relógio desenha mesmo um padrão gráfico em tempo real comparando essa carga com a do dia anterior.

### 2.2.3 – Obstáculos

Entra-se agora nos obstáculos à melhoria da eficiência energética, principalmente em infra-estruturas de larga escala para computação de elevado desempenho. É possível agrupar as maiores barreiras que impedem a execução de melhores práticas e a implementação de novas técnicas e tecnologias em várias categorias principais [12], sendo estas: crescimento na procura das TIC e consequentes centros de computação; medo de arriscar em novas implementações; falta de fundos para melhorias; incentivos nulos ou fracos; falta de conhecimentos e pessoal qualificado; falta de espaço nas instalações existentes; e necessidade de ferramentas diversas.

No crescimento da procura das TIC, o que acontece é que muitos utilizadores e organizações se começam a aperceber da grande capacidade de infra-estruturas de computação de elevado desempenho e começam a procurá-las e a usar cada vez mais as tecnologias de informação na resolução de problemas ou processamento de trabalhos, por exemplo. Esta maior procura origina por isso uma quase constante expansão no volume de dados ou de computação, necessidade de aplicações novas e com um alcance mais extenso e a mudança para novas gerações de equipamentos TIC. Muitas vezes é a dificuldade para identificar o equilíbrio entre dedicar tempo a novos serviços urgentes que produzem resultados imediatos e visíveis ou às actividades que proporcionam sustentabilidade a longo prazo por aumento da eficiência. Esta procura pelas TIC faz com que mesmo com preocupações de optimização do seu sistema, muitas organizações cheguem a um ponto em que pouco mais podem fazer ao nível de melhores práticas de forma a contra atacar o aumento de consumos originado por esta maior procura. Nestes casos, resta esperar que a indústria consiga avanços tecnológicos em *hardware*, *software* e equipamento para melhorar a eficiência energética.

O medo de arriscar explica-se facilmente pela realidade de que experimentar novas implementações traz consigo o risco de originar falhas na disponibilidade das infra-estruturas, o que é muito mais grave a nível empresarial, hoje em dia, do que não se economizar alguns kW/h de energia. Daí a grande hesitação em efectuar mudanças nas estruturas existentes.

A falta de fundos para melhorias e ausência de incentivos ou estes serem fracos explicam-se pelo simples facto de que para mudar e inovar, é preciso sempre capital inicial o que muitas das vezes é incomportável no orçamento das empresas. Também não há ainda grandes incentivos para organizações cujas infra-estruturas TIC sejam eficientes energeticamente, acontecendo que as organizações também não são obrigadas a pagar taxas por ter um consumo de potência considerado elevado.

Devido à constante necessidade de se educarem e se porem a par das novas técnicas rumo a mais economia energética, muitas empresas apontam a falta destes conhecimentos e pessoal especializado para não as pôr em prática, optando por não desperdiçar tempo com acções educativas deste tipo. Também a falta de novas práticas devidamente documentadas é um obstáculo à sua implementação por várias organizações.

A falta de espaço nas instalações é um facto limitador no caminho para o *Green Computing*, já que o equipamento para dar potência e para refrigerar as infra-estruturas de elevado desempenho computacional precisa de espaço suficiente e adequado para ser instalado.

Finalmente, a não utilização de ferramentas para monitorizar e gerir os grandes *datacenters* é mais um obstáculo. Muitas organizações não fazem uso das ferramentas já existentes hoje em dia, a nível de escalonadores e monitorizadores por exemplo, e não fazem sequer nenhuma medição do uso e eficiência de energia dos seus sistemas.

### 2.2.4 – Soluções/Plataformas para Green Computing

Para concluir o estudo sobre o principal tema deste trabalho, irá agora ser exposto um estudo efectuado sobre as soluções que já existem actualmente para ultrapassar os obstáculos mencionados e alcançar uma computação eficiente energeticamente e, por isso, mais ecológica, económica e proveitosa. Irão ser estudadas mais em detalhe duas abordagens que são de significativa relevância para a dissertação que será realizada posteriormente, por serem possíveis formas e pontos de partida para a solução e realização dos objectivos específicos desse trabalho. Depois será feita uma visão geral sobre restantes soluções existentes que são também importantes no contexto da eficiência energética no mundo das TIC em sistemas distribuídos de larga escala para computação de elevado desempenho.

**Modelo EARI**

A primeira abordagem é detalhada em [13]. Este modelo, denominado EARI (*Energy Aware Reservation Infrastructure*), tira partido das lacunas na utilização dos nós da *Grid*. A ideia geral é uma infra-estrutura que desligue os nós não usados e os ligue novamente quando são necessários para um utilizador. Este modelo ON-OFF assenta em três ideias principais: desligar nós não activos, prever a próxima reserva e consequentes recursos necessários e agregar reservas, relacionando-se com o escalonador da *Grid* e impondo-lhe decisões que sejam mais eficientes energeticamente.

Trabalhos anteriores mostram que as *Grids* operacionais não são normalmente utilizadas a 100% da sua capacidade. Em [13], ao analisar o caso da *Grid5000*, que é uma *Grid* experimental localizada em França para investigação em *Grid Computing*, observou-se que a percentagem real de utilização da *Grid* era de 50.57%. Baseado nestes factos, conclui-se que o consumo de energia pode ser reduzido quando a plataforma não é usada no seu máximo. A estrutura EARI tem de ser então capaz de: desligar nós não usados; prever o uso dos nós de forma a ligar os nós que são precisos num futuro próximo; agregar algumas reservas de recursos de forma a evitar ciclos ON/OFF frequentes.

A nível da arquitectura global do EARI teve-se em conta que em sistemas distribuídos de larga escala, para reduzir o consumo de energia deve-se actuar nos nós, nos dispositivos de rede e no escalonador. O EARI apresenta um algoritmo que será usado integrado com o escalonador para gerir os recursos da *Grid*.

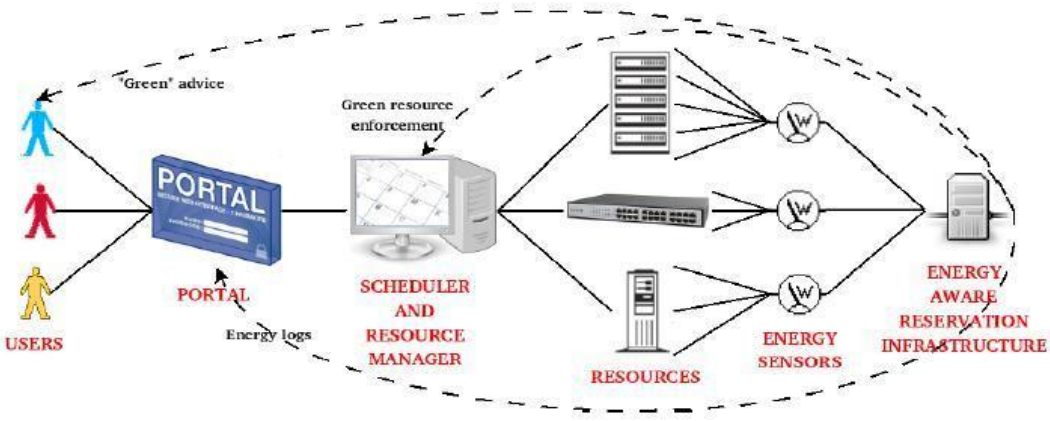



**Figura 2.1** – Arquitectura global da estrutura EARI [13]

A figura anterior ilustra a arquitectura geral da estrutura EARI. Cada utilizador faz uma reserva de recursos através de um portal, reserva essa tratada e validada pelo escalonador, que depois faz a gestão dos recursos, dando acesso a estes aos diferentes utilizadores que submeteram pedidos, de acordo com a agenda do escalonador. Há ainda sensores que medem parâmetros de energia dos recursos. Estes dados são recolhidos pela infra-estrutura e são usados para formar conselhos ecológicos que são enviados ao utilizador para influenciar as suas escolhas aquando da reserva de recursos. Por fim, a estrutura EARI ainda decide que recursos estão ligados e desligados.

De referir que neste trabalho [13] também se estudou o consumo de potência em alguns nós da *Grid5000*, nomeadamente na sua estrutura de Lyon. Os resultados obtidos mostram que no momento de ligação dos nós (ciclo *boot*) estes têm um grande consumo de potência, cerca de 300 a 400 Watts. Também têm um grande consumo enquanto estão em períodos de inactividade (ciclo *idle*), cerca de 190 Watts. Também no momento de desligar os nós (ciclo *turn off*) a potência consumida é significativa, sendo cerca de +20 Watts que em relação ao ciclo *idle*. Outras experiências mostraram que uma aplicação que exija um acesso intensivo ao disco pode gastar quase +10 Watts, uma comunicação de alta performance na rede entre +20 e +22 Watts e uma aplicação que exija um uso intensivo do CPU entre +20 e +26 Watts, tudo valores em relação ao consumo em estado ligado mas inactivo (ciclo *idle*).

O modelo EARI utiliza algoritmos de gestão de recursos. Antes de mais, é preciso esclarecer a definição dos componentes do EARI. Uma reserva é definida por (*l, n, t0*), em que *l* é a duração da reserva em segundo, *n* o número de recursos necessários e *t0* o instante inicial desejado para a reserva. Para que uma reserva seja válida tem de se ter *l* >> 1 e *n* <= *N* e *t0* >= *t*, onde *t* é o momento actual e *N* o número total de recursos que o escalonador controla. Ao aceitar uma reserva o escalonador insere-a na sua agenda que também contém

todas as reservas que ainda não começaram. Quando se iniciar, a reserva é movida para o histórico que contém assim todas as reservas passadas e actualmente em actividade.

O algoritmo utilizado na altura da submissão de uma nova reserva é dado pela expressão *R* = (*l, n0, t0*). Como é normal, neste instante inicial desta reserva específica haverá *n* recursos já em uso no *Cluster* ou *Grid*, portanto para ver se a reserva é passível de ser aceite ou não, tem de se verificar se *n0* <= *N* - *n*. Se não for possível iniciar a reserva no período desejado (*t0*), é calculado um novo instante inicial da reserva, agora *t1*, o mais cedo possível tendo em conta as reservas anteriores já presentes na agenda do escalonador. Depois, estima-se qual a energia que a reserva consumirá ao começar em diferentes instantes, sendo eles:

. em *t0* (ou *t1*, se *t0* não era possível);

. logo depois do próximo tempo final de uma reserva, tempo esse representado por *tend*;

. *l* segundos antes do próximo tempo inicial possível que é chamado *tstart*;

. durante um período de folga (*slack period*), em *tslack* (tempo maior ou igual a 2 horas e utilização da *Grid* ou *Cluster* inferior a 50%).

Para alcançar estas estimativas é então necessário calcular *t1* como visto anteriormente, *tend*, *tstart* e estimar *tslack*. O grande objectivo é agregar as reservas de modo a poupar energia evitando ciclos *boot* e *turn off*. Este modelo não impõe a solução melhor ao utilizador mas dá-lhe a escolher entre as várias soluções e o utilizador depois faz a opção que achar conveniente.

Na alocação de recursos do EARI para calcular *tend*, procura-se o próximo final de uma reserva na agenda e verifica-se se os recursos disponíveis a partir desse momento serão suficientes para começar *R* nesse instante. Se não for, repete-se o processo até se encontrar uma altura possível para começar a reserva. *tend* é assim definido como o tempo final da reserva encontrada. Já *tstart* é calculado procurando pelo instante inicial da próxima reserva na agenda e verificando se é possível colocar *R* antes desta. Para isso, é importante que este tempo inicial seja pelo menos em *t* + *l*, em que *t* é o instante actual e *l* a duração de *R*. Se não for possível, mais uma vez repete-se o processo até encontrar uma reserva na agenda que possibilite isto. Finalmente, estas estimativas sobre consumo de energia e tempos de início são enviadas ao utilizador que escolhe a solução que lhe agradar mais. A reserva é então inserida na agenda de acordo com a solução escolhida e o seu número passado ao utilizador. A abordagem do EARI permite então que o utilizador faça a reserva exactamente quando quer ou que por exemplo a atrase para permitir uma computação e alocação de recursos mais eficiente energeticamente. Para a alocação de recursos o escalonador tem ainda em conta o coeficiente de potência de cada um, escolhendo os recursos com maior coeficiente. Este é calculado através da média dos consumos energéticos de cada recurso calculada durante reservas sobre um grande período de tempo. Um coeficiente maior indica um recurso que gasta menos energia. Deste modo, quando o escalonador aceita na sua agenda uma nova reserva, atribui-lhe assim recursos que já estejam ligados e com prioridade os recursos com maior coeficiente de potência. A não ser que o utilizador escolha explicitamente determinados recursos, esta política é sempre aplicável.

O modelo EARI apoia-se ainda num algoritmo de libertação de recursos. Primeiro é calculado o consumo real total da reserva e dá-se a informação ao utilizador, guardando-a também no histórico para os algoritmos de previsão. Algumas definições de componentes do algoritmo são importantes na sua compreensão. Primeiro, o conceito de reserva *iminente* é uma reserva que começará em menos de *Ts* segundos em relação ao instante actual. O objectivo é obter *Ts* tal que seja o tempo mínimo que garanta uma poupança de energia se se desligar o recurso durante este tempo, ou seja, se for desligado determinado recurso *Ts* segundos serão poupados *Es* Joules, sendo *E* a grandeza energia. De referir que *Es* é uma energia fixa, sendo o mínimo de energia que se consegue reduzir facilmente. Ainda há uma definição de tempo específica, chamada *Tr*, que se relaciona com o tipo de recurso a desligar. Por exemplo, desligar um disco ou um computador é diferente de desligar uma placa Ethernet e *Tr* faz então essa distinção, que difere de recurso para recurso. Há ainda as grandezas *Pidle* (que se refere à potência consumida em Watts por um recurso ligado mas em estado inactivo, ou seja, em ciclo *idle*), *Poff* (que se refere à potência consumida em Watts por determinado recurso quando está desligado) e *Eon -> off* e *Eoff -> on* (que se refere à energia em Joules necessária para desligar e ligar um recurso, respectivamente), definições ilustradas na figura seguinte.

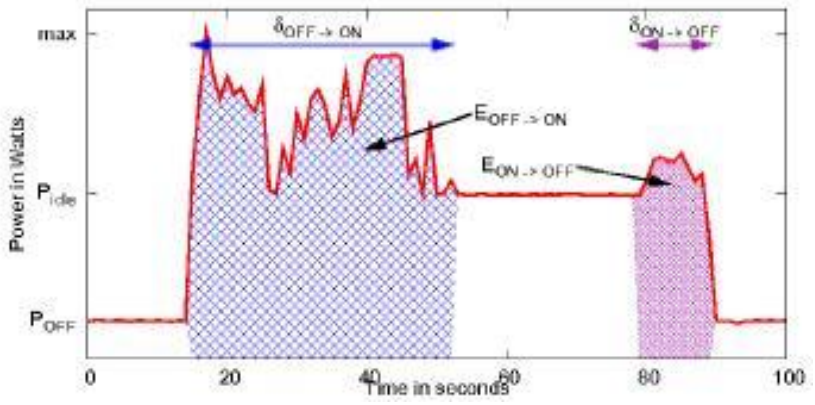



**Figura 2.2** – Processo de ligar e desligar um recurso, ilustrando as diferentes definições [13]

*Ts* define-se assim como:

$$Ts = ((Es - Poff * \sigma tot + Eon \to off + Eoff \to on) / (Pidle - Poff)) + Tr \quad (2.1)$$

sendo $\sigma tot = \sigma on \to off + \sigma off \to on$

Como se pretende que haja tempo suficiente para desligar e ligar o recurso, deve-se ter *Ts* – σtot >= 0. Se ao encontrar recursos já ligados e livres mas cuja próxima reserva é *iminente*, os recursos são deixados ligados num ciclo *idle*, já que durante este tempo se perde menos que *Es* Joules por recurso. A figura seguinte ilustra este caso.

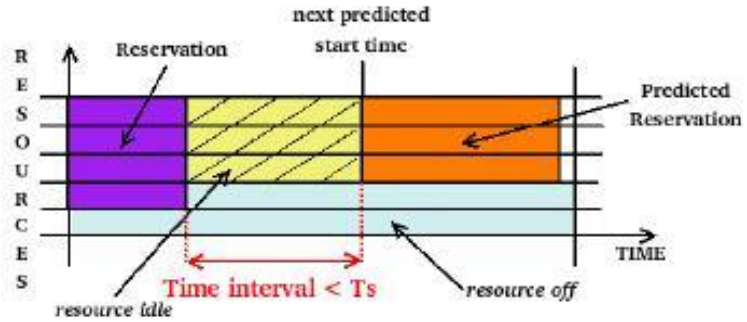



**Figura 2.3** – Exemplo de gestão de recursos mostrando o papel de *Ts* (recursos deixados em ciclo *idle*) [13]

Procura-se então outros recursos ligados que estejam à espera de uma reserva *iminente* previamente estimada. Agora, para estes *m* recursos encontrados, é necessário estimar quando ocorrerá a próxima reserva e quantos recursos necessitará para ser processada, reserva essa a que se chama *Re* = (*le, ne, te*). Se se verificar que *Re* não é *iminente*, podem ser desligados os recursos já que isso permitirá poupar mais energia do que se estes se mantiverem ligados até à próxima reserva. Este processo ilustra-se na figura seguinte.

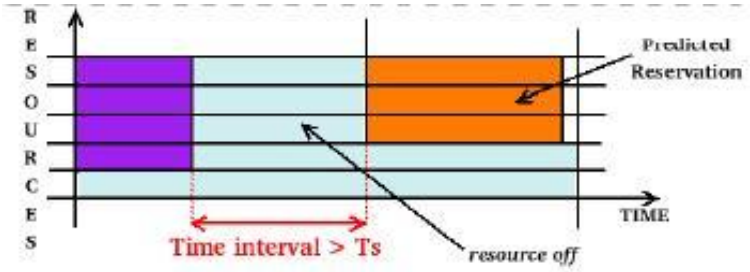



**Figura 2.4** – Exemplo de gestão de recursos mostrando o papel de *Ts* (recursos desligados) [13]

Se *Re* for *iminente*, determina-se o mínimo entre *m* e *ne* ( *min(m, ne)* ) para encontrar o menor número de recursos que estão livres em *te* durante *le* segundos por forma a poderem aceitar a reserva. Estes recursos são mantidos ligados durante *Ts* + *Tc* e desligam-se os outros. *Tc* é o tempo médio entre a submissão da reserva pelo utilizador e a sua aceitação pelo escalonador, incluindo tempo de cálculo das estimativas de energia e um tempo mínimo para responder ao utilizador. Adicionalmente, se houver recursos em ciclo *idle* e a reserva que acabou de chegar não for *iminente*, esses recursos podem ser desligados.

A juntar a estes algoritmos para gestão dos recursos, a estrutura EARI tem ainda um mecanismo de gestão da carga. A função deste mecanismo é assegurar que o escalonador não utilize sempre os mesmos recursos. Ao usar um modelo de topologia, permite conhecer a posição de diferentes recursos e assim distribuir geograficamente as reservas. Deste modo, os recursos activos não estão todos juntos uns aos outros o que faz com que haja menos produção de calor, o que origina menos trabalho e consumo para o sistema de refrigeração.

A eficiência do EARI consiste na capacidade de fazer previsões precisas, estimando a próxima reserva, a energia consumida por determinada reserva e a estimativa de um período

de folga (*slack period*) como visto anteriormente. No entanto, é conveniente manter o algoritmo de previsão suficientemente simples de forma a não perder muito tempo na sua computação o que atrasaria o trabalho do escalonador e o processo das reservas. Os algoritmos de previsão usados pelo EARI estão descritos em detalhe em [14]. O modelo EARI necessita prever três tipos de valores:

. a próxima reserva, ou seja, o seu instante inicial, a sua duração e o número de recursos que requer;

. a energia consumida por determinada reserva;

. quando ocorrerá o próximo período de folga.

A estrutura EARI tem tido bons resultados e boa recepção por parte da comunidade. Tem também servido como boa base de partida para ainda melhores formas de fazer esta gestão dos recursos de uma *Grid* ou *Cluster*.

**Gestão de potência baseada em Virtualização (máquinas virtuais)**

Outra solução encontrada para gerir a potência e os recursos de um sistema de *Grid Computing* é apresentada em [15]. Esta solução baseia-se num *hypervisor* (que é um monitor da máquina virtual) de máquina virtual *Xen* de forma que a solução é transparente para o utilizador.

Sucintamente, refira-se que o processo de virtualização cria então várias máquinas virtuais, cada uma com a mesma arquitectura do único computador que as comporta, o chamado *host*. Permite assim que se usem várias máquinas virtuais no mesmo computador e que cada uma use o seu próprio sistema operativo ou aplicações sem que entrem em conflito com as outras. Outro processo deste género, criado para colmatar as perdas de performance da virtualização devido a operações sensíveis, é a paravirtualização. Neste processo transporta-se explicitamente o sistema operativo para o *hypervisor*, o que permite criar arquitecturas nos domínios virtuais ligeiramente diferentes da do *host*, eliminando assim o efeito das operações sensíveis que reduzem a performance da virtualização. O *hypervisor* é um *software* que serve então não só para fazer uma repartição dos recursos entre os vários domínios virtuais mas também para aplicar um isolamento entre esses domínios. A virtualização e paravirtualização são as duas maneiras mais usadas de implementar este *software*.

Um exemplo das operações sensíveis mencionadas são as chamadas operações TLB (*Translation Look-aside Buffer*), que são usadas para diminuir o tempo de translação, por exemplo, das estruturas de dados como as tabelas de páginas. Para não usar tanta memória, estas tabelas estão muitas vezes apenas parcialmente guardadas na memória principal enquanto a grande parte da tabela está guardada num disco duro e é trocada da e para a memória quando é necessária. As operações TLB fazem cache das informações dessas tabelas para evitar este processo. [16]

A solução apresentada faz então uso da virtualização com a ferramenta *Xen*, sendo um sistema de colocação dinâmico para domínios virtuais numa *Grid*, implementando um *hypervisor* em cada nó da *Grid*. A ideia base é poupar energia concentrando as máquinas virtuais no menor número de nós possível, podendo os restantes nós serem desligados. O critério usado para o algoritmo de colocação é baseado na utilização do CPU. No entanto, pode ter adicionalmente outros critérios em conta como o tráfego de rede ou a utilização da memória. A ferramenta *Xen* tem um mecanismo de migração [17] que permite distribuir os domínios virtuais sem que haja grande interrupção de serviço. O uso de um servidor de ficheiros que exporta os sistemas operativos e dados para cada domínio virtual resolve o problema da migração de dados.

A arquitectura assenta num modelo cliente/servidor. Um domínio standard já preparado, denominado Domínio 0, corre em cada nó e tem uma ferramenta activa, chamada *harvesting agent* (agente de colheita) que faz a monitorização do uso dos recursos do nó e como estes são divididos pelos domínios virtuais desse nó, monitorizando e recolhendo informações sobre uso do CPU, memória e rede.

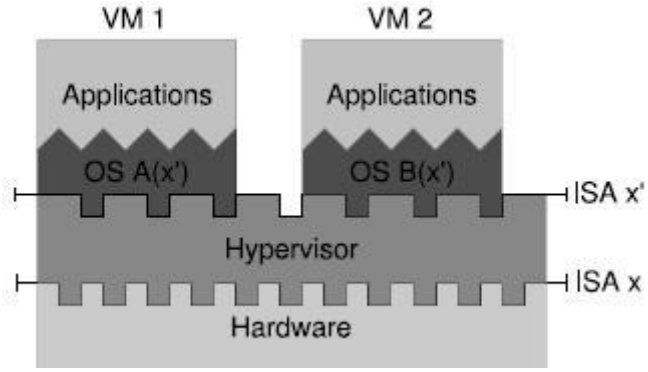



**Figura 2.5** – Arquitectura de um sistema virtualizado com a ferramenta *Xen* [15]

Cada um destes agentes envia ainda periodicamente relatórios ao agente decisor (*decision agent*). Estes relatórios contêm informações estatísticas sobre estado da memória, bytes enviados e/ou recebidos pela interface de rede virtual ou a utilização do CPU virtual. Estas informações ajudam a perceber a percentagem de CPU usado por cada máquina virtual e, por soma destas, o uso global de CPU por cada nó. O *harvesting agent* é também responsável por avisar o agente decisor de mudanças no estado de um nó. Quando um nó é parado manda uma notificação de saída ou se um nó estiver pronto a receber máquinas virtuais manda uma notificação de chegada. O agente decisor ao analisar estes relatórios consegue assim ficar com uma visão global da *Grid* e do uso dos seus recursos pelos seus domínios virtuais, permitindo ainda através das estatísticas e notificações referidas decidir se é viável aumentar os recursos para cada máquina ou o número de máquinas em cada nó. Um algoritmo final determina as acções que acontecerão em cada nó, como ligar, desligar ou migração de domínios virtuais.

O agente decisor executa na sua própria máquina e constrói duas vistas da *Grid* com base nos relatórios e notificações enviadas pelo *harvesting agent* presente no Domínio 0 de cada nó. Uma das vistas é orientada aos nós, ou seja, dá informações sobre o seu estado, se estão

ocupados ou livres e os seus endereços MAC e IP. Outra vista é orientada para os domínios virtuais, informando sobre a sua localização, os recursos que estão a consumir e qual o nível de saturação do host. Com estas informações o agente decisor tenta concentrar as máquinas virtuais necessárias no menor número possível de nós evitando, claro, a saturação de cada um deles. O algoritmo assenta em duas variáveis: os limiares (*thresholds*) de saturação e subcarga dos nós.

Com o objectivo de concentrar as máquinas virtuais sem sobrecarregar os nós, o algoritmo identifica o nó com menos carga de trabalho e encontra o domínio que consome mais energia de CPU neste nó. Posto isto, este domínio virtual é migrado para outro nó mais adequado. Supondo um sistema homogéneo, como um dos *Clusters* da FEUP, por exemplo, o consumo de CPU será praticamente idêntico no novo nó. Deste modo, o nó de destino é escolhido tendo em conta que a soma da carga que o domínio virtual está a processar com a carga total que o nó de destino tem em mãos não ultrapassa o limiar de sobrecarga do nó. Quando se encontrar um nó que viabilize este processo, a migração é feita. Este processo é repetido na próxima iteração se o nó ainda estiver subcarregado e houver um local mais apropriado para os domínios virtuais.

Outro perigo a evitar é o sobrecarregamento dos nós devido a um aumento do consumo de CPU por parte dos domínios virtuais, pois estes nem sempre fazem o mesmo trabalho e ao mudar este consumo pode variar. Para evitar esta degradação de performance, faz-se a migração do domínio com a menor carga para um nó que o possa acolher. Este nó de destino é agora escolhido com base no seu uso de CPU, que deverá ser o que tiver o máximo de consumo de CPU sem estar sobrecarregado, antes e depois da migração. Se nenhum nó estiver disponível, será ligado um nó que estava desligado anteriormente e o domínio virtual será então movido.

Esta solução apoiada na ferramenta *Xen* [15] que permite trabalhar na camada de abstracção por ela oferecida consegue então economizar energia devido a este processo de migração da computação, através de algoritmos de colocação de domínios virtuais. A concentração da computação é feita por migrar dinamicamente domínios virtuais considerando os seus requisitos a nível de recursos e o estado actual dos nós da *Grid* ou *Cluster*. Esta concentração de domínios virtuais no número mínimo de nós permite assim desligar nós não usados, sendo a computação mais eficiente energeticamente.

Por fim, será feita uma visão global sobre algumas soluções alternativas existentes para tornar a computação mais eficiente energeticamente e, portanto, mais ecológica e económica em infra-estruturas de larga escala para computação de elevado desempenho.

## IVS, CVS E VOVO

Como observado em [15], para a gestão de potência em sistemas únicos ou *Clusters* existem também políticas específicas como o IVS (*Independent Voltage Scaling*), o CVS (*Coordinated Voltage Scaling*) e o VOVO (*Vary On Vary Off*), políticas que apesar da sua relativa simplicidade são menos eficazes na poupança de energia em relação às soluções apresentadas anteriormente.

O IVS é uma política direccionada para os nós de computação. Cada nó ajusta a frequência e voltagem do seu processador para o valor mais baixo possível que permita manter níveis de performance considerados aceitáveis. Esta política é muito simples e não requer nenhum ambiente específico ou informações sobre aplicações a correr no nó.

O CVS é uma política global, ou seja, cada nó informa periodicamente um monitor da sua carga de trabalho. Avaliando a carga geral na rede, o monitor estima uma carga média e informa os nós deste valor. Os nós adaptam assim a sua voltagem e frequência a esta média. Este método é efectivo se a carga de trabalho for uniformemente distribuída pelos nós, caso contrário a sua efectividade é questionável.

O VOVO adapta dinamicamente o número de nós disponíveis de acordo com a carga de trabalho do *Cluster* ou *Grid*, fazendo uso de uma estratégia de concentração da carga. Um monitor estima a carga geral na infra-estrutura. A partir desta estimativa, o VOVO conclui quantos nós são precisos para processar essa carga e os nós não usados são desligados. O monitor pode ainda definir uma tolerância de performance para, se possível, sobrecarregar alguns nós para aumentar mais a eficiência energética. Este método pode ser muito eficaz principalmente se um *Cluster* estiver quase totalmente não usado. Contudo, este método necessita de uma tecnologia de migração de trabalhos, que pode ser implementada a vários níveis, o que é muitas vezes pouco viável de se fazer a nível da aplicação ou sistema operativo, uma vez que o problema de economizar energia é mais para as organizações que usam *Grids* ou *Clusters* e não para utilizadores a usar ou desenvolver aplicações.

### ÏANOS

A *Intelligent ApplicationN-Oriented Scheduling* (ÏANOS) [18] é uma estrutura construída sobre um *middleware Grid* (*middleware* ou mediador é, no domínio da computação distribuída, um *software* que permite fazer a mediação entre outros *softwares*. São exemplos as ferramentas *UNICORE* ou *Globus Toolkit*, que auxiliam na construção e implementação de Grids computacionais, tendo o objectivo de implementar comunicação e suportar a partilha de recursos em sistemas distribuídos através de vários processos e objectos num grupo de computadores). A estrutura ÏANOS tem como componentes um sistema de informação que guarda dados relativos aos recursos e aplicações, um agente (*broker*) de recursos ao nível da *Grid* que permite escolher os recursos mais apropriados para garantir a Qualidade de Serviço (QoS) exigida pelos utilizadores aquando da execução de qualquer aplicação, um metaescalonador que se liga ao *middleware Grid* subjacente e obtém informação sobre o estado dos recursos e ainda um sistema de monitorização que guarda o comportamento das aplicações aquando da sua execução.

Um dos conceitos pioneiros da estrutura ÏANOS é que as necessidades das aplicações são caracterizadas e estas são também parametrizadas, o mesmo acontecendo com os recursos, o que possibilita que o agente dos recursos saiba, por exemplo, qual o poder computacional de cada nó, bem como o estado da sua memória e performance de rede. A partir destas informações é possível prever qual o tempo de execução num dado recurso.

Outro aspecto original do ÏANOS é o algoritmo usado na função do agente de recursos: é a função correspondente entre o que a aplicação precisa e o que o recurso oferece com base num pedido de um nível determinado de QoS por um utilizador. Este algoritmo é assim baseado numa função de custo e tem em conta todos os custos da submissão de um trabalho

para a rede, tais como custo de execução, custo de tempo de espera, custo de transferência de dados, custo de licença e custo ecológico. A QoS especificada pelo utilizador pode ser, por exemplo: "Eu quero os meus resultados o mais rápido possível independentemente dos custos" ou "Eu quero os meus resultados pelo menor custo monetário possível, independentemente do tempo" ou uma mistura de ambos. O metaescalonador reúne todo o estado da *Grid* no momento da submissão de trabalhos e o custo geral é então calculado para todos os recursos, mediante um processo de minimização para todos os recursos e o trabalho é atribuído ao recurso melhor e mais eficiente energeticamente.

Assim, a eficiência energética da estrutura ÏANOS é alcançada por melhor uso dos recursos da *Grid*, baseando-se em três mecanismos principais. Primeiro, de acordo com as necessidades das aplicações escolhe o conjunto de recursos mais apropriados para a correr. Depois, atribui a aplicação e agenda-a para o recurso que seja mais eficiente energeticamente. Terceiro, por ter conhecimento prévio do que a aplicação precisa para ser processada, é possível ainda mudar a arquitectura alvo, desligando por exemplo um processador para aumentar a eficiência sem reduzir a performance.

Um protótipo da estrutura ÏANOS foi já desenvolvido, implementado e testado numa plataforma de teste internacional conjunta entre Alemanha e Suíça no contexto da *CoreGRID Network of Excellence* e do projecto *Swiss ISS*. Ainda se planeia implementar esta estrutura para a *European e-Infrastructure* por meio de um novo projecto Europeu.

### Aproveitamento do calor residual

O aproveitamento do calor residual originado pelos *datacenters* pode ser um factor de melhoria na eficiência energética destas infra-estruturas. De facto, a reutilização deste calor desperdiçado pela computação efectuada nestes centros pode ser usado para minimizar a sua emissão de dióxido de carbono (CO2), que como já referido anteriormente, constitui uma porção bastante relevante das emissões globais.

Como as infra-estruturas de computação de elevado desempenho consomem grandes quantidades de potência eléctrica, consequentemente produzem calor em excesso que é dissipado para o ambiente. Também hoje em dia ainda se usam fontes de energia primárias como óleo e gás para aquecimento de espaços. O conceito explicitado em [19] oferece uma perspectiva atractiva do ponto de vista ecológico e económico, fazendo uso desse calor em excesso de forma produtiva.

Concomitantemente, quase 50% da energia consumida e das emissões de CO2 de um *datacenter* refrigerado por sistemas a ar é causada não pela computação mas pelos sistemas de refrigeração necessários para evitar que os componentes microelectrónicos destas infra-estruturas sobreaqueçam e falhem.

A resposta a esta realidade é um sistema de refrigeração baseado em líquidos, de forma a atingir uma computação com alta performance e pouco consumo de potência.

A água é um óptimo agente de refrigeração. Esta é capaz de capturar calor até 4000 vezes mais eficientemente que o ar. Se forem usados líquidos refrigeradores de alta performance, água com temperaturas até cerca de 60º C permitiriam manter os processadores a operar a menos do máximo de 85ºC. Isto elimina a necessidade extrema de energia por parte dos

sistemas de refrigeração na grande parte dos *datacenters* actuais. Adicionalmente, o calor residual pode ser reutilizado para aquecimento de espaços.

Com estas considerações em mente, Meijer, Brunschwiler, Paredes e Michel [19] construíram um pequeno protótipo na *IBM* como plataforma de teste para um novo supercomputador que será entregue ao *Swiss Federal Institute of Technology Zurich (ETH)*. Este supercomputador terá um sistema de refrigeração a água inovador, sendo esperado uma redução na ordem dos 40% a nível do consumo de energia e uma diminuição das suas emissões de CO2 na ordem dos 85%. Esta grande redução nas emissões de CO2 é possível pois o calor em excesso produzido será usado para aquecer edifícios da Universidade.

## 2.3 – Clusters

### 2.3.1 – Conceito

O aparecimento de problemas e projectos que necessitam de cada vez mais poder computacional na sua resolução levou a que se começasse a pensar em soluções alternativas para o uso de apenas computadores simples para resolver cada problema. Nos anos 80 do século passado três tendências começaram a ser seguidas e desenvolvidas [34]: microprocessadores de alta performance, redes de alta velocidade e ferramentas padronizadas para computação distribuída de alto desempenho. Culminando este período, Donald Becker e Thomas Sterling iniciaram um projecto de um sistema de processamento distribuído. O seu protótipo era constituído por uma aglomeração de computadores pessoais, não especializados e baratos para computação paralela. Em 1994 criaram então o primeiro *Cluster* deste tipo, o projecto *Beowulf* [35].

Assim, pode-se já inferir qual o conceito de *Cluster*. Este é uma aglomeração de estações de trabalho ou computadores pessoais, conectados entre si e que fazem computação paralela, permitindo a coordenação de esforços computacionais para a partilha e processamento de uma tarefa comum entre múltiplos processadores como se fossem um único processador. Em muitos dos casos actuais, os computadores constituintes dos *Clusters* são de alta performance e a rede que os liga é de baixa latência e grande largura de banda, o que permite então oferecer serviços rápidos e fiáveis no processamento de aplicações com um consumo computacional intensivo. Este sistema é uma alternativa viável e mais barata aos supercomputadores, apresentando um poder computacional semelhante e muitas vezes superior aos supercomputadores. Como referido por Baker, Buyya e Hyde [36], é aceite que um *Cluster* de estações de trabalho (*workstations*) de alta performance consegue competir com os melhores supercomputadores que a *IBM* ou a *SGI* oferecem, sendo que uma organização pode montar um cluster deste tipo por cerca de $50000 enquanto a construção de um supercomputador custaria cerca de $200000, ou seja, quatro vezes mais. Ainda segundo Baker e Buyya [38], um *Cluster* é capaz de oferecer maior fiabilidade do que os supercomputadores e, se for bem concebido, ainda maior tolerância a falhas.

### 2.3.2 – Componentes Típicos

Um *Cluster* compreende todos os componentes presentes em qualquer LAN com computadores pessoais ou estações de trabalho, como processadores, memórias, placas de rede, *middleware*, sistemas operativos, cablagem, ferramentas e ainda outras utilidades. A arquitectura dos *Clusters* pode no entanto variar drasticamente. Alguns reutilizam sistemas mais antigos possivelmente usados originalmente em escritórios por exemplo. Outros são montados a partir de material comercial adquirido em lojas normais da especialidade. Outros ainda são construídos à volta de processadores SMP (*Symmetric MultiProcessing*) e tecnologias de rede personalizadas. Também a configuração física dos *Clusters* varia, incluindo desde alguns computadores pessoais localizados numa sala de aulas por exemplo a pilhas de *motherboards* armazenas e conectadas numa sala de serviços de computação. Pela variedade possível dos seus componentes e arquitecturas, o número de aplicações que podem ser executadas num sistema de *Clusters* é enorme e tem uma tendência crescente. Por exemplo, estes sistemas suportam não só aplicações paralelas de alta performance como computação para investigação em campos científicos, como também aplicações comercias como um servidor Web de alta performance de carga balanceada como *HotBot*, que usa uma base de dados *Oracle* paralela.

### 2.3.3 – Implementações

Na lista anual da organização *Top500* [39] aparecem normalmente listados muitos *Clusters* entre os melhores supercomputadores mundiais.

O supercomputador *System X* da *Virginia Tech*, que em 2006 foi considerado o vigésimo oitavo supercomputador mais potente do mundo é um *Cluster* com 12.25 TFlops de velocidade que compreende 1100 máquinas *Apple XServe G5* de 2.3 GHz e duplo processador cada [37].

O projecto *Stone Soupercomputer* adopta o conceito explicitado dos sistemas *Beowulf* [37].

A *Sun Microsystems* tem uma especificação, a *JavaSpaces*, que permite, pela via de uma memória distribuída partilhada, que se faça um *Cluster* de computadores.

Há assim muitas arquitecturas e implementações possíveis para formar *Clusters*, com as vantagens em relação ao uso singular de computadores pessoais ou de supercomputadores anteriormente referidas. De referir que os *Clusters* são sistemas homogéneos, ou seja, as máquinas que os compões e os componentes são iguais, além de que normalmente estes sistemas estão confinados a uma organização, aumentando o seu poder computacional. No entanto, é comum hoje em dia ver sistemas de *Clusters* interligados criando um sistemas de maior escala, heterogéneo e com ainda maior poder computacional, como é o caso dos sistemas de *Grid Computing* e, no caso da FEUP, do GridFEUP que interliga vários *Clusters*.

## 2.4 – Grid Computing

### 2.4.1 – Origem

Na última década houve um aumento significativo nos sistemas de computação de alta performance e no desempenho das redes de computadores. Isto deriva essencialmente do facto de se ter melhorias substanciais a nível de *hardware* e *software*. Este avanço tecnológico permitiu desenvolver sistemas de elevado desempenho computacional com baixo custo, como os *Clusters*, para resolver problemas que necessitam de um processamento de dados e número de recursos elevados, em vários domínios de aplicação. No domínio da ciência, o melhor desempenho computacional de sistemas deste tipo permitiram aos cientistas alargar o escopo das suas investigações, podendo processar mais parâmetros que nunca e a melhoria das redes permitem a partilha de dados e resultados experimentais quase instantaneamente entre vários lugares do globo. Assim, a colaboração entre várias entidades é fomentada e melhorada e ajuda a que projectos de vários domínios sejam mais completos e tenham maior alcance, qualidade e rapidez de execução.

Hoje em dia, a génese de programas que facilitam a criação e exploração de estruturas orientadas para atacar projectos e problemas de larga escala dá-se a um ritmo bastante elevado. Estes programas são, colectivamente, designados de *eScience* [20].

Estes progressos na performance das redes e poder computacional das estruturas de computação provocam que os dados gerados, processados e analisados no contexto dos programas *eScience* sejam em número elevado e bastante distribuídos geograficamente. Devido a tudo isto, estes ambientes enfrentam vários desafios no que concerne à gestão, ao acesso, à distribuição, ao processamento e armazenamento destes dados. Consequentemente, criou-se uma infra-estrutura que juntasse recursos distribuídos de grandes áreas como bases de dados, redes de alta velocidade, supercomputadores e *Clusters*, dando origem ao que hoje é comummente conhecido como *Grid Computing*.

Embora este conceito estivesse a ser congeminado há já algum tempo, foi o livro de Ian Foster e Carl Kesselman lançado em 1998, *"The Grid. Blueprint for a new Computing Infrastructure"* [21], que revolucionou o campo da computação oferecendo-lhe um novo domínio de pesquisa: o *Grid Computing*. As ideias originais contidas no livro fazem o paralelo entre uma *Grid* computacional e a rede eléctrica *(electrical grid)*, que oferece acesso transparente, consistente, fidedigno e universal à energia eléctrica, como se quer em sistemas de *Grid Computing*. Embora a definição precisa de *Grid Computing* seja ainda hoje motivo de discussão, esta comparação é globalmente aceite.

As principais motivações para o aparecimento do *Grid Computing* são: a necessidade de se processar grandes quantidades de dados; redução de custos, sendo uma alternativa mais barata relativamente aos supercomputadores; a possibilidade do uso de recursos geograficamente dispersos e portanto aumento de cooperações multidisciplinares entre várias organizações; necessidade de aumento do desempenho na resolução de problemas de larga escala; disponibilidade de recursos de forma consistente e fiável; a partilha de recursos e/ou serviços; oportunidade de aproveitar os ciclos ociosos (ciclos *idle*) dos recursos.

### 2.4.2 – Conceito

O conceito de *Grid Computing* ainda não é totalmente consensual nos dias de hoje pois pese embora a ideia básica não tenha mudado muito desde a sua definição por Foster e Kesselman [21], há muitas pessoas com diferentes ideias sobre o que uma *Grid* realmente é.

No sentido de tentar obter uma definição mais precisa do que é uma *Grid* e o processo de *Grid Computing*, Stockinger elaborou um trabalho [22] cuja base assentou num inquérito enviado para mais de 170 investigadores de *Grid* de todo o Mundo. Este inquérito baseava-se na seguinte directriz:

"*Try to define what are the important aspects that build a Grid, what is distinctive, and where are the borders to distributed computing, Internet computing etc.*" [22]

Portanto, nesta sondagem aos investigadores de *Grid* pretendia-se que cada um deles desse a sua definição sobre os aspectos importantes que constituem uma *Grid*, o que é distintivo e quais as fronteiras em relação à computação distribuída, Internet *Computing*, etc.

Houve mais de 40 pessoas a responder ao inquérito, o que pese embora poder haver diferentes visões sobre o assunto em questão, oferece uma perspectiva realista sobre se há uma percepção mais ou menos comum do que é uma *Grid* ou se há muitas opiniões divergentes. Como referido por Stockinger no seu artigo [22], o inquérito mostra que embora várias pessoas descrevam características da *Grid* de forma diferente, há muitas sobreposições nessas descrições e quase nenhumas contradições, o que permite concluir que os resultados do inquérito são coerentes e há um conceito geral de *Grid* globalmente aceite e presente entre os diferentes investigadores.

**Conceito Global**

A ideia geral de *Grid Computing* instaurada por Foster e Kesselman [21] não mudou mas foram adicionadas mais algumas ideias complementares. Por exemplo, Gregor von Laszewski do Argonne National Lab dos Estados Unidos da América diz que "na visão de *Grid* há uma distinção entre (a) *a abordagem* ou o paradigma da *Grid*, que representa um conceito e ideia geral para promover uma visão para sofisticadas e internacionais colaborações científicas e orientadas a negócios e (b) a *instanciação física* de uma *Grid* de produção baseada em recursos e serviços disponíveis para possibilitar a visão para sofisticadas e internacionais colaborações científicas e orientadas a negócios"1.

De acordo com o trabalho de Foster e Tuecke [23] referido em [22], uma estrutura de *Grid Computing* deve providenciar um conjunto de capacidades técnicas como as seguintes:

. "Modelação dos recursos. Descreve os recursos disponíveis, as suas capacidades e as relações entre eles para facilitar a sua descoberta, provisionamento e gestão da qualidade de serviço;

. Monitorização e notificação. Providencia visibilidade para o estado dos recursos - e notifica aplicações e serviços de gestão da infra-estrutura de mudanças no estado - para possibilitar a descoberta e manter qualidade de serviço. O registo de eventos significativos e transições do estado também é preciso para auxiliar funções de contabilidade e auditoria;

. Alocação. Assegura qualidade de serviço através de um conjunto inteiro de recursos no tempo de vida da sua utilização por uma aplicação. Isto é possibilitado negociando o(s) nível(eis) de serviço e assegurando a disponibilidade dos recursos apropriados através de uma qualquer forma de reserva - essencialmente, a criação dinâmica de um acordo ao nível dos serviços;
. Provisionamento, gestão do tempo de vida e desmantelamento. Possibilita que um recurso alocado seja configurado automaticamente para o uso de uma aplicação, gere o recurso durante a tarefa em mãos e repõe o recurso no seu estado original para uso futuro;
. Contabilidade e auditoria. Segue a utilização de recursos partilhados e providencia mecanismos para transferir custos entre comunidades de utilizadores e para cobrar pelo uso de recursos por aplicações e utilizadores."
. Juntando a estas, Greg Astfalk da HP (*Hewlett Packard*), dos Estados Unidos, refere que a segurança é também uma capacidade importante no âmbito de sistemas de *Grid Computing*.

Analisando as várias respostas dadas a Stockinger [22], é possível inferir que o conceito mais aceite de *Grid Computing* é o de uma infra-estrutura que permite efectuar computação distribuída através de vários domínios administrativos, ou seja, é a combinação de vários sistemas distribuídos e de alta performance, desempenho e fluxo de dados para que haja uma partilha, colaboração e coordenação distribuída de recursos que pertencem a diferentes entidades ou organizações.

Stockinger [22] refere ainda as definições oferecidas pela *CoreGRID Network of Excellence* e da *GGF*, que são comummente aceites na área da computação. Quanto à *GGF* [25], há a seguinte definição para um sistema de *Grid Computing*:

"Um sistema que se interessa com a integração, virtualizaçao, e gestão dos serviços e recursos num ambiente distribuído, heterogéneo que suporta colecções de utilizadores e recursos (organizações virtuais) através de domínios administrativos e organizacionais tradicionais (organizações reais)."1

Quanto à *CoreGRID Network of Excellence*:

"Uma infra-estrutura totalmente distribuída, reconfigurável dinamicamente, escalável e autónoma para providenciar um acesso independente do local, difusivo, confiável, seguro e eficiente a um conjunto coordenado de serviços encapsulando e virtualizando recursos (poder computacional, armazenamento, instrumentos, dados, etc.) por forma a gerar conhecimento."1

Estas definições têm traços semelhantes àquelas aceites e divulgadas por Buyya e Venugopal em [24], como se vê por exemplo na definição do Globus Project, um dos vários projectos mundiais com o âmbito de desenvolver *Grids* para diferentes propósitos a diferentes escalas, que diz:

"Uma infra-estrutura que possibilita o uso integrado e colaborativo de computadores terminais, redes, bases de dados e instrumentos científicos pertencentes e geridos por múltiplas organizações."1

Ainda neste trabalho [24], refere-se que há várias estruturas de *Grid Computing* que se baseiam numa arquitectura baseada em Organizações Virtuais (OV) [26], como já referido em

definições anteriores. Estas não são mais que organizações reais que se juntam numa colaboração multiadministrativa e, muitas vezes, multidisciplinar, no sentido de partilhar recursos e colaborarem para atingir um determinado objectivo comum. Uma OV define que recursos os seus membros têm disponíveis e estabelece um conjunto de regras de acesso e utilização dos mesmos, tendo em conta prioridades e objectivos próprios de cada OV.

Importa ainda ressalvar que uma *Grid* pode ser construída com diferentes tecnologias, não havendo assim uma única tecnologia para esse efeito, até devido às diferentes necessidades e objectivos das organizações que adoptam sistemas de *Grid Computing*. Existem alguns *softwares* e serviços que ajudam a construir *Grids*, como se verá mais à rente neste estudo, permitindo conectar de forma mais fácil e intuitiva fontes de dados e recursos computacionais distribuídos de forma a suportar a análise e colaboração distribuída. Estas ferramentas (de *software* ou serviços) são normalmente designadas de *middleware Grid*. Por exemplo, tecnologias baseadas em serviços Web são usadas para construir *Grids* mas outras tecnologias podem ser utilizadas. De acordo com John Morrison da University College Cork da Irlanda, "(...) uma multiplicidade de tecnologias é até desejável e pode ser necessário usá-las concomitantemente numa *Grid* heterogénea" por forma a potenciar a sua performance.

**Características de um sistema de Grid Computing**

As características e capacidades que uma infra-estrutura de *Grid Computing* deve ter e que reúnem a concordância da grande maioria dos investigadores desta área são [22]:

. Colaboração: Uma *Grid* deve então partilhar os recursos de uma forma distribuída e justa. Esta colaboração, se for feita de modo apropriado, poderá resultar em vantagens emergentes e sinergias positivas entre utilizadores e provedores de serviço, inalcançáveis de outra forma.

. Agregação: A *Grid* permite agregar a capacidade de recursos individuais originando um recurso virtual de maior e muito alta capacidade, sendo preservada a capacidade do recurso individual. Assim, isto permite processar aplicações e trabalhos maiores mais rapidamente sendo que de um ponto de vista local se mantém a capacidade de correr novas aplicações. Os recursos devem poder ser adicionados dinâmica ou estaticamente.

. Virtualização: Os serviços de *Grid Computing* são muitas vezes apresentados numa interface que esconde a complexidade dos recursos adjacentes. Portanto, a *Grid* tem a capacidade de virtualizar a soma de todos esses recursos num recurso singular, permitindo por exemplo um ponto de entrada único para submissão de trabalhos ou para correr aplicações de larga escala. A virtualização pode cobrir tanto os recursos computacionais como os dados. Rodrigo Fernandes de Melo da Universidade de São Paulo, Brasil, refere, como mencionado em [22], a lista de recursos a virtualizar : Pode-se ter a utilização da *Grid* como virtualização de fluxo de trabalho, ou seja, o uso de serviços da *Grid* para executar e gerir processos através de várias plataformas; pode haver uma *Grid* de dados como virtualização de dados, ou seja, a gestão de conjuntos de dados partilhados independentemente do sistema de armazenamento onde eles estão guardados; ou ainda uma *Grid* semântica como virtualização da informação, tendo assim a capacidade de processar e "debater" sobre atributos de vários repositórios independentes de

informação. O mesmo autor refere no mesmo trabalho [22] outros tipos mais específicos de virtualização que um ambiente de *Grid Computing* deve ter.

. Orientação do Serviço: As *Grid* providenciam serviços de computação aos seus utilizadores, seguindo a ideia de uma arquitectura orientada aos serviços.

. Heterogeneidade: A *Grid* é normalmente um sistema heterogéneo, compreendendo na sua composição componentes de *hardware* e *software* diferentes, com diferentes performances.

. Controlo Descentralizado: Uma *Grid* deve fazer uso de mecanismos de controlo distribuídos, já que os recursos são propriedade de diferentes entidades, sendo esta uma das dificuldades destes sistemas, o facto de não haver um proprietário único para o sistema global, num todo.

. *Standardização* e interoperabilidade: Uma *Grid* tem a necessidade de cada vez mais integrar distintas tecnologias e elevar os acordos na *standardização* dos serviços. Esta promove definições *standard* da interface para serviços que precisam de interoperar para criar a infra-estrutura distribuída que consiga realizar as tarefas que os utilizadores querem fazer.

. Transparência de acesso: Através da virtualização dos recursos nos sistemas de *Grid Computing* é possível oferecer níveis de transparência no acesso do utilizador comum aos serviços e recursos da *Grid*, ou seja, sem este ter de ter presente a arquitectura adjacente ou topologia da rede.

. Escalabilidade: Mesmo que um problema novo não seja resolvido pelas infra-estruturas ou implementações *Grid* usadas, é muitas vezes a escala dos dados, recursos e utilizadores que contribui para a maior complexidade da *Grid*.

. Reconfigurável : Como referido na definição da *CoreGRID*, a *Grid* deve ser reconfigurável dinamicamente.

. Segurança: Uma característica basilar da *Grid* é a segurança no acesso aos recursos. Esta é uma problemática que se apresenta aos utilizadores da *Grid* sempre que a usam e é assim essencial que as infra-estruturas de *Grid Computing* consigam garantir segurança nos acessos aos recursos que se estendem por vários domínios administrativos.

### 2.4.3 – Arquitectura Geral

As características e capacidades mencionadas anteriormente são cruciais para permitir que uma infra-estrutura de *Grid Computing* possa resolver problemas e efectuar tarefas científicas, de engenharia ou de negócios diversos. Os vários componentes da *Grid* que possibilitam as características mencionadas estão normalmente organizados em camadas. A figura seguinte mostra a pilha de *hardware* e *software* numa arquitectura típica de uma infra-estrutura de *Grid Computing*.

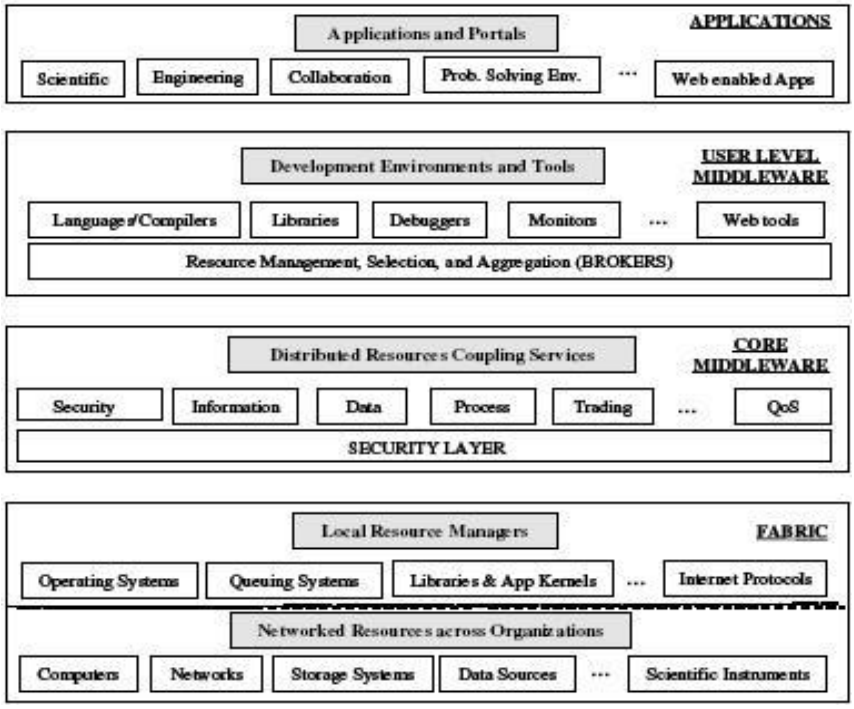



**Figura 2.6** – Arquitectura Grid em camadas e seus componentes [24]

Há assim quatro camadas [24]. A camada *Fabric* consiste essencialmente nos recursos disponíveis na estrutura, como computadores, redes, armazenamento e instrumentos científicos. Múltiplas arquitecturas de recursos computacionais podem ser ligadas e funcionar de modo interoperável como *Clusters*, servidores ou supercomputadores. Outros instrumentos científicos oferecem o acesso e transmissão de dados em tempo real entre vários pontos do sistema ou para sistemas de armazenamento.

A camada *Core middleware* oferece uma virtualização da camada *Fabric* providenciando um método consistente para aceder aos recursos da *Grid*. Os serviços que esta camada oferece vão desde alocação de recursos, registo e descoberta de informação, segurança, acesso a sistemas de armazenamento, gestão de processos remota e aspectos de Qualidade de Serviço como reserva de recursos.

A camada *User-Level middleware* utiliza as interfaces disponibilizadas pela camada anterior para providenciar abstracções e serviços de mais alto nível. Incluem-se nestes ambientes de desenvolvimento, ferramentas de programação e agentes que gerem os recursos e fazem o escalonamento de aplicações para executar nos recursos globais.

A camada *Applications and Portals* providencia serviços como serviços de aplicações baseadas em serviços Web, onde os utilizadores podem submeter e obter resultados dos seus trabalhos através da Web em serviços remotos. Também oferecem aplicações para, por exemplo, simulação de parâmetros, que necessita de grande poder computacional e acesso a dados remotos. Os serviços desta camada são desenvolvidos a partir das interfaces da camada anterior.

Observa-se que cada camada constrói os seus serviços a partir das interfaces oferecidas pela camada anterior.

### 2.4.4 – Iniciativas de Grid Computing

Há um grande interesse nas tecnologias de *Grid Computing* actualmente devido às possibilidades que sistemas deste tipo oferecem no aumento da eficiência da computação distribuída e, consequentemente, na resolução de problemas e tarefas de larga escala. Os projectos relacionados com o *Grid Computing* podem seguir um objectivo de implementar infra-estruturas *Grid* que podem ser usadas por investigadores e utilizadores de vários campos nas suas pesquisas e tarefas ou ter o objectivo de criar *middleware* e investigar o desenvolvimento de *software* e mecanismos que ajudem a optimizar a *Grid*. Serão apresentados alguns dos projectos mais importantes deste tipo a nível dos Estados Unidos e continente Europeu, sendo que muitos deles são motivados pelo aparecimentos de novos problemas e projectos de larga escala que envolvem produção e tratamento de dados a uma escala enorme e cada vez maior, o que origina a necessidade de distribuição da computação e análise dos dados para uma execução mais rápida e eficiente [24].

Nos Estados Unidos várias plataformas de teste de *Grids* de produção foram já implementadas sobre infra-estruturas *Grid* como a US NSF (*United States National Science Foundation*). Uma das maiores plataformas de teste é a *Grid3* que cobre 25 locais através dos Estados Unidos e Coreia, locais esses que são usados em conjunto para executar aplicações de áreas como Biologia ou Astronomia. Outras plataformas existentes incluem a GriPhyN (*Grid Physics Network*), PPDG (*Particle Physics Data Grid*) ou SDSS (*Sloan Digital Sky Survey*), entre outras. A nível de pesquisa e desenvolvimento de *software* e mecanismos para optimizar a *Grid* existe uma ferramenta muito importante, a *Globus Toolkit* [24] [40] da *Globus Alliance* liderada pelo Argonne National Laboratory [41], que ajuda na construção e implementação de *Grids* operacionais. Outros exemplos são o projecto *Condor*, da Universidade de Wisconsin, Madison, para mecanismos de computação de grande fluxo de dados e o *AppLeS* da Universidade da Califórnia, San Diego, para aplicativos de escalonamento.

Na Europa existem também várias iniciativas associadas ao Grid Computing. O programa iniciado no Reino Unido em 2001, *eScience* e o projecto *Data Grid*, fundado pela União Europeia, que foi agora sucedido pelo EGEE (*Enabling Grids for E-sciencE*) são talvez as duas maiores iniciativas de implementação de *Grids* operacionais no continente Europeu [24]. O projecto *eScience* focou o seu objectivo em promover e aumentar o envolvimento do Reino Unido na implementação e provisão de estruturas *Grid* e investigação e desenvolvimento de ferramentas e mecanismos para facilitar a construção destas infra-estruturas. O projecto EGEE tem praticamente os mesmos objectivos mas numa escala mais alargada, no sentido de querer providenciar infra-estruturas e *middleware Grid* aos cientistas para os auxiliar nos seus projectos e investigações. Também o CERN [42] (*European Organization for Nuclear Research*) e a comunidade HEP (*High-Energy Physics*) se juntaram no estabelecimento de um projecto designado *International Data Grid project* que visa estabelecer uma rede de investigação para desenvolver tecnologia *Grid* (estruturas e *middleware Grid*), demonstrar a eficácia das *Grids* através de experiências em plataformas de teste implementadas e demonstrar a capacidade de usar componentes de baixo custo para implementar estes sistemas.

Devido ao aparecimento de cada vez mais *middleware* e *software* associado ao *Grid Computing*, a definição de *standards* universais é crucial para garantir a interoperabilidade entre diferentes estruturas deste género. A GGF [25] tem reunido esforços no sentido de

consolidar as melhores práticas e mecanismos associadas ao *Grid Computing* através de *standards* ou normas por forma a permitir o rápido crescimento e adopção da computação distribuída [24] [25].

Pesando os factos, constata-se que o *Grid Computing* e a computação distribuída é uma área em expansão e evolução. O aparecimento de projectos a escalas nunca antes vistas torna esta computação distribuída importantíssima para optimizar o processamento de quantidades massivas de dados inerentes a esses projectos.

## 2.5 – Cloud Computing

### 2.5.1 – Conceito

Nos dias de hoje, como já referido anteriormente, o mundo da computação tem vindo a afastar-se da política do "faça você mesmo" e direcciona-se para o campo da computação distribuída e partilhada. O *Cloud Computing* segue esta tendência, sendo que uma definição mais simples para esta forma de computação seja que permite aceder a ficheiros, programas, dados e serviços de "terceiras" entidades que estão sob o domínio de uma outra "terceira" entidade através da Internet, pagando só pelos recursos e serviços usados. Tipicamente, os provedores de infra-estruturas de *Cloud Computing* oferecem diferentes aplicações e serviços através da Internet que são acedidas via um browser Web no computador dos utilizadores, enquanto o *software* e dados são armazenados em servidores. Portanto, toda a estrutura da *Cloud* é transparente para o utilizador comum. Oferece assim recursos, dinamicamente escaláveis e muitas vezes virtualizados, como serviço através da Internet. O termo *Cloud* é usado por comparação com a nuvem que se usa muitas vezes para designar a Internet, no sentido de ser uma abstracção de toda a arquitectura e estrutura adjacente necessária para a implementação da estrutura mas que é transparente para o utilizador.

Uma *Cloud* tem três características essenciais distintivas em relação às infra-estruturas tradicionais de armazenamento e processamento de dados e serviços [29]. Primeiro, é um serviço *on-demand*, ou seja, a facturação da sua utilização é feita pelo tempo que se usa o serviço, normalmente ao minuto ou à hora, e o utilizador só paga pelos recursos que usa. Em segundo lugar, o *Cloud Computing* oferece uma infra-estrutura elástica, no sentido que um utilizador pode usar tantos recursos como os que quiser em cada utilização, umas vezes mais outras menos, consoante a sua conveniência. Em terceiro lugar, é o provedor da *Cloud* que gere e suporta totalmente o(s) serviço(s). Assim, o utilizador não tem de ser proprietário da estrutura física necessária para executar os serviços por si pretendidos, mas sim aluga uso a uma terceira entidade, o provedor da *Cloud*, precisando normalmente apenas de um computador pessoal e uma ligação à Internet.

Uma *Cloud* pode ser privada ou pública. Se for pública quer dizer que os seus serviços estão disponíveis a qualquer utilizador da Internet. Se for privada significa normalmente que é uma estrutura oferecida numa rede privada ou *datacenter* de uma organização e que providencia serviços a um número limitado de pessoas com acesso a essas redes[29].

Os avanços em tecnologias de virtualização e computação distribuída, bem como as melhorias a nível do desempenho da rede Web mundial fizeram disparar o interesse na área do *Cloud Computing*.

A arquitectura típica de uma *Cloud* [31] envolve *software* e *hardware* concebido por um arquitecto de *Cloud* que normalmente trabalha para um provedor de *Cloud Computing* [28] [29]. Normalmente, os vários componentes, cada um com a sua função, comunicam uns com os outros através de APIs (*Application Programming Interfaces*), normalmente serviços Web, de forma a trabalharem em conjunto sobre uma interface universal. Esta arquitectura estende-se até ao cliente, enquanto os browsers Web e os serviços de *software* acedem às aplicações da *Cloud*. O armazenamento numa *Cloud* é flexível, permitindo que os nós de dados sejam escaláveis ao evitar o uso de servidores de dados centralizados.

### 2.5.2 – Vantagens e Preocupações

As seguintes vantagens podem então ser identificadas para sistemas de *Cloud Computing* [27]:

. Os utilizadores não precisam de fazer um grande investimento inicial em recursos, normalmente necessitando apenas de um computador e ligação à Internet para usufruir dos serviços oferecidos pelo *Cloud Computing*. Recursos como servidores, sistemas de armazenamento, de rede e *software* são todos geridos pela terceira entidade provedora da *Cloud*, bem como custos com electricidade, manutenção e até pessoal necessários para executar os serviços;

. A elasticidade dos serviços de *Cloud Computing* é um atributo bastante apreciado, sendo que os utilizadores podem usar tantos recursos computacionais ou serviços quanto desejarem em cada momento;

. Este serviço é provavelmente mais barato que manter os recursos em sua propriedade, já que os utilizadores só pagam pelos recursos que usam pelo tempo que os usam, alugando assim uso dos recursos e serviços à terceira entidade e não tendo que os manter, poupando nos custos associados a essa gestão;

. Com um computador e ligação à Internet, os utilizadores podem aceder aos serviços oferecidos pelos provedores de *Clouds* em qualquer altura e em qualquer lugar.

Estas vantagens tornam o *Cloud Computing* uma abordagem bastante interessante e apelativa para a computação no futuro. No entanto, há também preocupações no que concerne à adopção de sistemas deste tipo. Alguns dos maiores receios identificados [27] são os seguintes:

. Disponibilidade: Em relação à disponibilidade, a maior preocupação é sobre o que acontece quando um provedor de *Cloud* fecha o seu negócio, por falência ou qualquer outra razão, ou deixa de estar capaz de fornecer os serviços que o utilizador foi levado a crer que teria disponíveis. Isto leva a que seja necessário uma escolha cuidada do provedor de *Cloud* e que haja sempre um plano alternativo caso esta situação aconteça. Ainda relativo a este tema, na grande maioria das vezes só teoricamente é que é possível ter 100% de disponibilidade do serviço, sendo necessário ter também planos de contingência se a falha, ainda que momentânea, no serviço seja incomportável para o utilizador. Por exemplo, os utilizadores poderão, como medida de precaução, não armazenar dados que são críticos para o seu

objectivo na *Cloud* ou mesmo usar uma *Cloud* adicional de apoio, como *backup*, ou manter versões das aplicações nos seus recursos pessoais para precaver situações em que a rede está em baixo e seja preciso trabalhar *offline*;

. Segurança e Privacidade: Em qualquer sistema de computação distribuído, a segurança e privacidade dos dados é uma questão essencial. O *Cloud Computing* não foge a esta regra, até sobretudo por ser executado via Internet, hoje em dia inundada de perigos como vírus ou piratas informáticos, que podem destruir os dados ou mesmo aceder aos computadores dos utilizadores ou seus dados pessoais se não forem tomadas medidas adequadas. É necessário então que os provedores de *Clouds* adoptem as ferramentas e métodos de segurança mais sofisticados e actualizados existentes actualmente, para providenciar o máximo de segurança possível aos utilizadores e fazer com que este receio diminua;

. Suporte/Apoio : Os utilizadores muitas vezes precisam de apoio especializado na resolução de problemas. Hoje em dia, os utilizadores comuns de SaaS (*Software as a Service*, serviço de *Cloud Computing* que será explicitado mais à frente) livre são normalmente deixados "sozinhos", não havendo qualquer tipo de apoio. As empresas normalmente pagam para ter este tipo de suporte para resolução de problemas. Portanto, é um facto que os provedores de *Clouds* têm de contratar e treinar pessoal especializado para oferecer melhor suporte e ajuda aos seus clientes, até para superar o suporte que os utilizadores estão acostumados com a computação tradicional (com os recursos e serviços alocados localmente);

. Interoperabilidade : A interoperabilidade refere-se à capacidade de migrar e integrar facilmente aplicações e dados entre provedores de *Clouds*. Esta é uma preocupação ainda não muito abordada no *Cloud Computing* mas existe e possivelmente ganhará força à medida que crescer o mercado para sistemas deste tipo.

### 2.5.3 – Formas e Provedores de Serviço

No trabalho de Kim [27] as formas de serviço identificadas são as seguintes: Serviços geridos, SaaS, serviços Web, computação utilitária (também chamada de IaaS em [29]) e PaaS (*Platform as a Service*). O que faz destes serviços "serviços *Cloud*" é o facto de os utilizadores poderem utilizá-los via Internet.

Um serviço gerido é basicamente a disponibilização de uma aplicação a uma empresa e não a um utilizador terminal directamente. Actualmente, há vários serviços deste tipo que podem disponíveis através da Internet, como é o caso de serviços de segurança (IBM, SecureWorks), serviços de *scan* para virus ou adware, serviços de anti-spam para emails, etc.

O serviço de SaaS é um mercado extenso. O conceito é que o provedor de *Cloud* suporta e fornece *hardware* da infra-estrutura e o produto em *software*, fazendo a interacção com o ciente através de m portal via Internet. O cliente beneficia assim apenas da funcionalidade do produto via Internet. Para os utilizadores terminais comuns existem serviços SaaS como o *Microsoft Windows Live* (que inclui o *Hotmail*, *Messenger* e *Photo Gallery*), o *Google Apps* (que inclui o *Gmail*, *Google Talk*, *Google Calendar*, *Google Docs*, etc) ou o *Zoho Office*. A nível empresarial alguns destes provedores de SaaS incluem a *salesforce.com* (para aplicações de vendas), o *Citrix* (para aplicações de apoio a reuniões), etc.

Os serviços Web são similares aos Saas[27]. Os provedores de serviços deste tipo fornecem APIs que ajudam os desenvolvedores no desenvolvimento de aplicações. Exemplos de serviços deste género são o *Google Maps*, o *U.S. Postal Service*, *Bloomberg*, etc, que fornecem APIs deste tipo.

A computação utilitária ou IaaS é um serviço recente que se baseia na disponibilização de recursos através de servidores virtuais com endereços IP únicos e blocos de armazenamento *on-demand*. Os clientes usam a API fornecida pelo provedor para iniciar, parar, aceder e configurar os seus servidores e unidades de armazenamento virtuais. Alguns provedores de serviços deste tipo são a *Amazon*, *Sun Microsystems*, *IBM*, *AT&T*, *Nirvanix*, *Hatsize*, *Joyent*, etc.

PaaS é definido normalmente como uma variação do SaaS, sendo um conjunto de *software* e ferramentas de desenvolvimento do produto hospedado na infra-estrutura do provedor da *Cloud*. Os desenvolvedores criam aplicações na plataforma do provedor através da Internet, portanto PaaS oferece um ambiente de desenvolvimento de aplicações. Provedores de serviços deste género incluem a *Google* (*Google App Engine*), *Yahoo* (*Pipes*), *Coghead* e *Dapper.net*.

## 2.6 – Escalonador

O acesso aos recursos necessários para executar os trabalhos e consequentes operações e processos computacionais inerentes tem de ser controlado de forma a que o uso e acesso a esses recursos seja eficaz. Outra vantagem deste controlo é permitir o uso ordenado, programado e justo dos recursos. Para isso, é usado um Escalonador (*Workload Manager*). O grande propósito de um sistema deste tipo é maximizar a eficiência do uso de recursos de uma maneira conveniente. Os utilizadores querem ter um acesso fiável e seguro aos recursos de forma a ter os seus pedidos e trabalhos atendidos e processados o mais rapidamente possível. Os administradores querem ter o maior conhecimento possível do estado dos nós dos sistemas e possíveis problemas para melhor entenderem o fluxo de trabalhos e os recursos disponíveis.

No caso da FEUP, recorre-se ao *software Moab* [43 - 45] para escalonar a atribuição de recursos dos *Clusters* da rede *Grid* da Faculdade para a concretização dos trabalhos submetidos, embora também seja usado o programa *gLite*, mas em menor escala.

O *Moab* é um escalonador altamente avançado desenvolvido pela *Cluster Resources, Inc.* direccionado a grandes estruturas computacionais como *Clusters* ou *Grids*. Sendo um escalonador, ele não só permite gerir a fila de trabalhos submetidos e os nós computacionais dos clusters, por exemplo, mas como que dita regras à rede dizendo aos recursos quais deles é que serão usados, para que trabalho e quando, sendo que se desejado tem ainda a opção de dizer por quanto tempo vão estar activos. Este sistema permite uma computação altamente adaptativa, permitindo que os recursos computacionais sejam facilmente configuráveis de acordo com mudanças ao nível da necessidade de uso ou devido a defeitos ou falhas, permitindo que sistemas ou nós com anomalias sejam automaticamente corrigidos ou substituídos. As características do *Moab* oferecem também possibilidade de observar estatísticas do cluster para análise de desempenho e extensivo diagnóstico deste, bem como aumenta a qualidade de serviço e a disponibilidade de recursos do sistema, permitindo então aos administradores um grande controlo sobre quais os trabalhos que são elegíveis para

serem escalonados e assim inseridos na fila de trabalhos, como esta fila é gerida e os trabalhos priorizados e em que nós serão os trabalhos processados. Portanto, o *Moab* é único em vários aspectos, já que atribui trabalhos aos diferentes nós da rede *Cluster*/*Grid* e é dinâmico, muitas vezes mudando o fluxo de trabalhos para melhor corresponder à disponibilidade do *Cluster*/*Grid* ou o contrário, adaptando o *Cluster*/*Grid* ao fluxo de trabalhos em fila de submissão.

As capacidades do *Moab* estão assentes em vários mecanismos internos que permitem melhorar a performance geral do escalonamento de trabalhos e alocação de nós para efectuar esses trabalhos, permitindo que os utilizadores corram mais trabalhos no mesmo sistema e que estes sejam finalizados mais rapidamente. Os mecanismos mais relevantes são reservas em avanço, especificações de Qualidade de Serviço e a chamada abordagem *backfill*. Ao nível das reservas em avanço, o *Moab* permite que um determinado sistema ponha de lado certos recursos durante um dado período de tempo para usos individualizados. Ou seja, assim é determinado que os recursos reservados são destinados a trabalhos com um fim específico ou a um utilizador específico. O acesso a estas reservas é controlado por uma lista de controlo de acesso especializada para estas reservas. O mecanismo de especificação da Qualidade de Serviço permite a utilizadores que tenham acesso especial a esta opção de especificar certos parâmetros de Qualidade de Serviço que queiram que seja satisfeita aquando do processamento do seu trabalho. Os benefícios deste mecanismo variam entre alta prioridade dos trabalhos, acesso a recursos adicionais ou isenção de certas políticas limitativas da configuração do *Moab* usada na organização. A abordagem *backfill* é uma característica do *Moab* que permite que certos trabalhos sejam processados fora da ordem em que estão na fila, desde que isso não atrase os trabalhos com maior prioridade na fila. De forma a aquilatar se um trabalho vai ou não ser atrasado por correr outro fora de ordem, cada um dos trabalhos submetidos deverá fornecer uma estimativa de quanto tempo necessitará para ser processado, a chamada janela de processamento. De notar que é bom que se estime este tempo por excesso, já que o *Moab* pode estar configurado para terminar trabalhos após o fim da janela de processamento. No entanto, este excesso na estimativa não deve ser demasiado grande pois assim atrasaria outros trabalhos alocando recursos inutilmente, o que vai contra o princípio de optimização da rede apregoado e implementado pelo *Moab*.

Em conclusão, o *Moab* permite que as organizações que o usam diagnostiquem e optimizem o uso dos seus recursos, nomeadamente ao nível de estruturas de computação elevada como *Clusters* ou *Grids*, visualizando ainda o comportamento usual dos seus utilizadores. Pode ainda ser integrado com sistemas de monitorização, bases de dados, gestores de identidade, gestores de licenças, outras redes e sistemas de armazenamento, proporcionando deste modo uma visão completa e coesa do *Cluster*/*Grid* organizacional. O *Moab* possibilita também a redução do uso de energia através de políticas inteligentes de gestão de potência e gestão dos trabalhos nos sistemas de computação das organizações, sendo assim uma inteligente e sofisticada ferramenta para decisões. Na FEUP, usa-se em simultâneo com o escalonador *Moab* o sistema de monitorização *Ganglia*, cujas características e funcionamento geral será referido seguidamente. Tudo isto ajuda então as organizações a atingirem os seus objectivos de uma computação de elevado desempenho mais eficiente e mais preocupada com questões ambientais, protegendo-as mesmo de eventuais normas e legislações que possam surgir que limitem a capacidade das infra-estruturas de computação de larga escala.

## 2.7 – Monitorizador

Na actualidade, a necessidade de ferramentas de monitorização eficientes tem crescido grandemente em consonância com o crescimento dos grandes *datacenters*, numa onda inversa à redução de pessoal administrativo neste *datacenters*. No entanto, desta necessidade decorre uma questão essencial que é o que cada um entende por monitorização. Enquanto o utilizador comum de uma organização, por exemplo, com uma estrutura *Grid* associada e que corre aplicações e trabalhos nos seus nós quer saber quando o seu trabalho vai ser processado ou quando ficará feito, um engenheiro do sistema pensará de que forma pode melhorar o serviço, como está este a decorrer actualmente e qual a performance das máquinas associadas. Ainda outras perguntas haverão, dependendo de quem as colocar e de qual o objectivo dessa pessoa ou do grupo a que ela pertence. Algures neste infindável rol de questões há várias soluções disponíveis para as funções de monitorização que se quer fazer, seja em *software open-source* ou comercial.

A grande dificuldade será então adoptar e implementar uma instalação e configuração de um *software* de monitorização que garanta a satisfação dos nossos objectivos. Isto deve-se ao facto de que nenhuma ferramenta monitorizará tudo o que se quer da forma pretendida, já que diferentes utilizadores terão diferentes necessidades de monitorização. Por isso, poderá ter de haver lugar a grandes alterações de configuração do *software* conforme as metas pessoais de cada pessoa ou organização, já que cada ambiente é único.

No meio desta miríade de ofertas é usado na FEUP o *software Ganglia* [46] [47], como solução integrada com o escalonador Moab, para monitorização da estrutura de alta computação da FEUP. Esta é uma ferramenta muito usada a nível de estruturas de computação elevada como *Clusters* ou *Grid*, tendo qualidades que a tornam adequadas até a outras tecnologias como *Cloud Computing*. O *Ganglia* é um sistema de monitorização distribuído escalável baseado num design hierárquico orientado para estruturas de *Grid Computing*, por exemplo, desenvolvido na Universidade da Califórnia, Berkeley. Aproveita tecnologias amplamente usadas como *XML* para representação de dados, *XDR* para transporte compacto e portável de dados e *RRDtool* para armazenamento e visualização de dados, usando ainda algoritmos e estruturas de dados desenvolvidas com grande detalhe e cuidado de forma a atingir alta concorrência e baixo overhead por nó. O grande princípio de operação do *Ganglia* é então recolher e juntar métricas (que podem ser velocidade do processador, uso da memória, etc) e analisá-las e controlá-las ao longo do tempo, sendo que no passado precisava de correr um agente em cada computador para recolher estas informações mas actualmente estas métricas podem ser obtidas a partir de quase qualquer nó graças ao mecanismo de spoofing do *Ganglia*. Este sistema tem também agentes escaláveis integrados nos hosts de destino, o que dota esta ferramenta de grande escalabilidade e a torna apetecível para grandes *datacenters* como *Clusters* ou *Grids*, sendo esta uma das

características chave do design original e intencional do *Ganglia*. Estas métricas retiradas do sistema operativo vão para um *host* de destino, que depois pode exibi-las ou passá-las em forma condensada para o nível superior, o que reflecte o seu design hierárquico e consequentemente a sua escalabilidade.

# Capítulo 3

# Normas e Parâmetros Energéticos

## 3.1 – Introdução

Os computadores são hoje uma parte importantíssima no quotidiano das pessoas, sendo que o cada vez maior poder computacional e armazenamento e circulação de informação, aliado a uma rede global de comunicações como a Internet, permitiram uma melhoria substancial da qualidade de vida. Isso é provado devido à actual oferta de serviços como as ferramentas de trabalho que os computadores oferecem, aliados a serviços como *homebanking*, *home shopping*, acesso a informação sobre os mais variados temas, etc. Esta performance computacional crescente permite também evolução e novas descobertas e experiências em campos tão variados como a medicina, astronomia, física, etc.

A possibilitar este armazenamento e troca de informação, dados e produtos estão inúmeros *datacenters* espalhados pelo mundo, estruturas de computação de elevado desempenho como *clusters* ou *Grids*, que albergam qualquer coisa como 20 milhões de servidores à escala mundial [48]. Devido ao facto de normalmente estas estruturas ficarem a correr todo o dia, 7 dias por semana, elas requerem uma quantidade enorme de electricidade.

Com os avanços tecnológicos até à data ao nível do *chip*, este foi ficando mais barato e com maior poder computacional. Isto teve o efeito de os consumidores requererem cada vez maior e melhor performance computacional, sendo que os produtores de servidores e processadores mundiais tiveram de oferecer essa performance, relegando assim para segundo plano a eficiência energética dos seus produtos. O mesmo se verificou nos proprietários de *datacenters*, cuja preocupação primária é o poder computacional, não se importando com os gastos energéticos e efeitos no ambiente do seu sistema.

Recentemente, com o aumento do preço da electricidade, os custos energéticos inerentes a estruturas deste tipo começaram a pesar bastante no orçamento geral dos seus proprietários, cortando mesmo os lucros das empresas ou companhias que usam estes sistemas, o que fez com que o problema da eficiência energética fosse posto na lista de prioridades ao lado da performance computacional. Também o aparecimento de algumas normas, *standards* e parâmetros energéticos aprovadas por governos ou instituições com significado neste meio, como a *Green Grid* [6] ou a EPA [49] (*Environmental Protection Agency*), por exemplo, levaram a esta mudança na forma de pensar das pessoas que possuem, usam ou gerem *datacenters*.

São alguns destes parâmetros, normas e *standards* que serão abordados neste capítulo.

## 3.2 – Gasto de Energia nos *Datacenters*

De acordo com o trabalho de Joe Loper e Sara Parr [48] de Janeiro de 2007, estudos recentes mostram que operações de computação, incluindo de processador, memória e fontes de alimentação internas dos computadores ou servidores são responsáveis por cerca de 33% a 75% dos gastos energéticos dos *datacenters*. O resto da energia utilizada pelos *datacenters* é maioritariamente gasta pelos equipamentos de refrigeração usados para remover o calor residual gerado nestas estruturas. De referir que este pode vir de várias fontes, como por exemplo da iluminação, do uso dos servidores, do uso dos processadores, das fontes de alimentação ou do equipamento auxiliar como equipamento de distribuição de potência fora do servidor, que usam energia eléctrica que acaba eventualmente como calor residual, que por sua vez tem de ser tratado pelos sistemas de refrigeração. Pode-se ver pela figura seguinte as áreas do *datacenter* que tipicamente mais gastam energia.

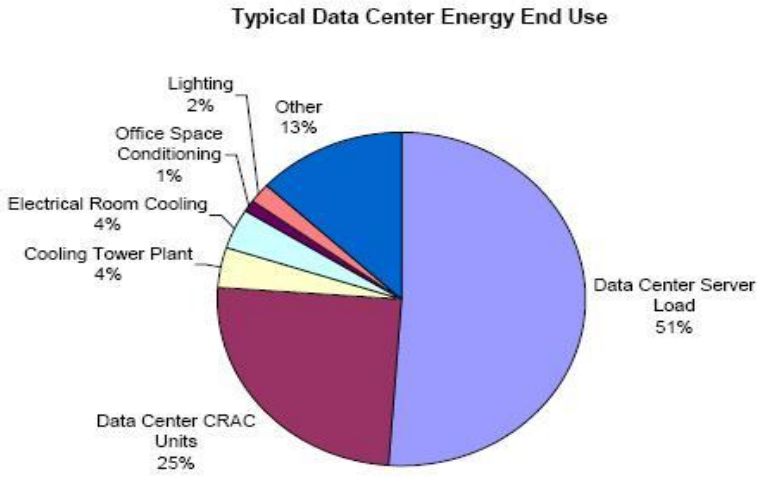



**Figura 3.1** – Áreas que tipicamente consomem mais energia nos *datacenters* [48]

Como se observa pela figura 3.1 e também como afirmado em [48], os sistemas de refrigeração, compreendendo as torres de refrigeração, os refrigeradores das salas de geração

de potência e as unidades CRAC (*Computer Room Air Conditioning*), que incluem aqui ar condicionados, *chillers*, ventoinhas, desumidificadores e bombas de refrigeração, podem ser responsáveis por 25% a 50% ou mais da electricidade consumida pelos *datacenters*.

Outro estudo [48] [50] revela também que o equipamento de distribuição de potência fora do servidor, como transformadores, UPSs (*Uninterrupted Power Supply*), inversores AC/DC, etc, pode constituir cerca de 11% dos gastos energéticos destas estruturas, podendo a iluminação do *datacenter* ascender a um peso de 2% no consumo de electricidade.

Por tudo isto, e pela enorme "pegada" ecológica deixada pelos cada vez maiores e mais poderosos *datacenters* a nível global, é importante haver *standards* e parâmetros energéticos apropriados pelos quais estes sistemas se devem reger.

## 3.3 – *Standards* e Parâmetros Energéticos Apropriados aos *Datacenters*

Esta é ainda uma área em expansão e a grande maioria dos *standards* e parâmetros não são ainda "leis" pelas quais os grandes *clusters*, *Grids* ou sistemas de *Cloud Computing* se devem restringir. Ainda não existe, por exemplo, um sistema de penalizações para quem não cumpre determinados parâmetros ou *standards* ou uma bonificação para um *datacenter* que os cumpra. Também ainda não existe uma entidade reguladora única que estabeleça regras de eficiência energética a que os *datacenters* tenham de obedecer, havendo isso sim algumas organizações, certo é que com um peso e aceitação crescentes, que vão tentando lançar e implementar *standards* no sentido de uma computação mais eficiente energeticamente, casos por exemplo da Green Grid [6], da ASHRAE [51] (*American Society of Heating, Refrigerating and Air-Conditioning Engineers*) ou da GIPC [52] (Green IT Promotion Council).

### 3.3.1 – Consumos de Potência Regulados pela EPA [49]

A EPA, trabalhando em conjunto com o Departamento de Energia dos EUA, criou um programa, o ENERGY STAR, que ajuda os consumidores a poupar nos gastos energéticos e consequentemente a poupar dinheiro, e que ajuda na protecção do ambiente, através de produtos e práticas eficientes energeticamente.

Os resultados deste programa são já visíveis, e estima-se que com a ajuda do ENERGY STAR, a população Americana tenha poupado cerca de 17 bilhões de dólares nas suas contas referentes a diferentes serviços, enquanto evitou a emissão de gases de efeito estufa num valor equivalente a 30 milhões de carros. [53]

O ENERGY STAR é apoiado e certificado pelo governo para reduzir o impacto ambiental do uso ineficiente de energia e para tornar mais fácil aos consumidores identificarem produtos que lhes permitem poupar nos gastos (energéticos e monetários) sem comprometerem o

desempenho destes e as suas características.

Para os produtos serem elegíveis a merecer um selo ENERGY STAR têm de satisfazer os requisitos de eficiência energética estabelecidos nas especificações dos produtos lançadas pela EPA.

No caso particular deste trabalho, interessa saber as especificações para computadores e servidores, que é onde a metodologia de eficiência energética a idealizar e implementar irá aplicar-se. Assim, com este facto em mente, analisou-se as especificações da ENERGY STAR para computadores [54] e servidores [55].

### *Computadores*

A versão 5.0 da especificação para computadores [54] da EPA cobre os seguintes produtos:

. Computadores Pessoais (*desktop computers*);
. Computadores Pessoais Integrados (*Integrated Desktop Computers*);
. Computadores Portáteis (*Notebook Computers*);
. *Workstations*;
. Consolas de Videojogos;
. Servidores de Pequena Escala;
. *Thin Clients*.

Dos produtos mencionados, estudo realizado focar-se-á nos Computadores Pessoais e Computadores Pessoais Integrados já que são estes tipos de produtos que podem estar presentes no *Cluster* ou *Grid* da FEUP ou em plataformas de teste destes sistemas onde o projecto incide (estruturas de computação de elevado desempenho). Antes de se apresentarem as especificações de acordo com a versão 5.0 dos requisitos do programa ENERGY STAR para computadores [54], há alguns termos relevantes para este trabalho cuja definição de acordo com a EPA deve ser apresentada anteriormente, para uma boa compreensão das especificações.

Um computador é um dispositivo que realiza operações lógicas e processa dados e que é composto, no mínimo, por uma unidade de processamento (CPU), instrumentos de entrada de dados do utilizador como teclado, rato, etc e um ecrã ou *display* para saída de dados. Incluem-se nesta definição dispositivos como unidades computacionais estacionárias ou portáteis, como computadores pessoais (*desktop computers*), computadores portáteis (*notebooks*), servidores de pequena escala ou workstations. De seguida, faz-se a distinção entre computadores pessoais e computadores pessoais integrados.

- Computador Pessoal (*desktop computer*): Computador cuja unidade principal se destina a um uso num local estacionário e permanente, sendo usado para uma grande variedade de aplicações em casa ou em escritórios. Utilizam um ecran, rato e teclados externos.

- Computador Pessoal Integrado (*Integrated Desktop Computers*): Sistema *desktop* no qual o computador e o ecrã de computador a ele associado funcionam como uma unidade única que recebe a sua potência AC através de um único cabo. Pode ser de duas formas: ou o ecran e o computador estão fisicamente ligados formando uma unidade única; ou o ecrã ser externo ao computador mas ligado a este por um cabo de alimentação DC, e tanto o computador como o

ecran recebem a energia através de uma única fonte de alimentação. Têm a mesma função dos Computadores Pessoais.

Outras definições relevantes:

1. Modo *off* (*off mode*): Refere-se ao modo em que o computador se encontra quando está no modo mais baixo de consumo de potência e que não pode ser influenciado/desligado pelo utilizador. Pode persistir por um período indefinido de tempo quando o dispositivo se encontra ligado à fonte principal de electricidade. Corresponde ao nível S5 do Sistema de Níveis ACPI [8] para sistemas onde os *standards* ACPI se aplicam.

2. Modo *sleep* (*sleep mode*): Um estado de baixo consumo energético que o computador pode entrar ou automaticamente, após um período de inactividade, ou por selecção manual. A partir de um certo evento despertador (*wake event*) - como conecções de rede ou uma acção do utilizador, por exemplo o uso de periféricos externos como o rato ou o teclado - o sistema pode acordar rapidamente. Para sistemas onde os *standards* ACPI se aplicam, corresponde ao nível S3 do Sistema de Níveis ACPI [56].

3. Estado *idle* (*idle state*): Estado em que a máquina está ligada, o sistema operativo (OS) está completamente disponível para utilização e a actividade está limitada às aplicações que o sistema faz por defeito. Ou seja, é um estado em que a máquina estando ligada e instantaneamente operacional, não está a fazer trabalho "útil".

4. Estado Activo (*busy state*): Estado em que o computador está a executar trabalho "útil" devido a um *input* de um utilizador ou a uma instrução através da rede. Aqui incluem-se funções de memória, procura de dados armazenados, processamento de dados diversos, ou cache, incluindo o tempo em estado *idle* em que espera pela próxima instrução do utilizador e antes de entrar em modos de baixa potência.

5. *Discrete* GPU (*Discrete Graphics Processing Unit*): Um processador gráfico com uma interface de controlador de memória local e uma memória local específica para gráficos.

6. Consumo Típico de Energia (TEC): Este é um método para testar e comparar a eficiência energética das máquinas. O método incide sobre a electricidade normalmente utilizada por um produto durante um determinado período de tempo enquanto está em operação normal. Para Computadores Pessoais (Integrados ou não) o critério principal é o valor de electricidade gasta anualmente, medido em kWh, usando medidas de níveis de potência médios dos modos operacionais escalados por um modelo assumido de utilização típico (*duty cicle* assumido).

7. Evento Despertador (*Wake Event*): É um evento que faz com que o computador saia de um estado *sleep* ou *off* e entre no estado activo. Pode ser desencadeado por uma acção do utilizador, uma acção escalonada ou um estímulo externo. Exemplos de um evento deste tipo pode ser um movimento do rato, toque no teclado, um evento de rede, etc.

Seguidamente apresentam-se os requisitos de eficiência e performance do programa ENERGY STAR[53] da EPA [49] para Computadores Pessoais e Computadores Pessoais

Integrados. Estes aparelhos são divididos em categorias, sendo elas as seguintes:

**Tabela 3.1** – Diferentes categorias de Computadores Pessoais e Computadores Pessoais Integrados definidas pela EPA

| | |
|---|---|
| Categoria A | Todos os computadores que não cumprem os requisitos necessários para serem incluídos nas categorias B, C ou D, serão incluídos na categoria A segundo a qualificação da ENERGY STAR; |
| Categoria B | Para se qualificarem nesta categoria, os computadores têm de ter:<br>- 2 *physical cores*;<br>- 2 GB ou mais de memória RAM. |
| Categoria C | Para se qualificarem nesta categoria, os computadores têm de ter:<br>- Mais do que 2 *physical cores*;<br>E ainda, adicionalmente, têm de ter pelo menos 1 das 2 características seguintes:<br>- Um *Discrete* GPU; e/ou<br>- 2 GB ou mais de memória RAM |
| Categoria D | Para se qualificarem nesta categoria, os computadores têm de ter:<br>- 4 ou mais *physical cores*;<br>E ainda, adicionalmente, têm de ter pelo menos 1 das 2 características seguintes:<br>- Um *Discrete* GPU com uma Largura de *Frame Buffer* maior do que 128 bit;<br>- 4 GB ou mais de memória RAM. |

De referir que um *Frame Buffer* é um dispositivo de output que permite armazenar e transferir para o ecrã os dados de uma *frame* de imagem completa. É um dispositivo de memória especializada (memória de imagem, memória de vídeo ou de tela). [57]

O TEC será determinado a partir da seguinte fórmula:

$$E_{TEC} = (8760/1000) * (P_{off} * T_{off} + P_{sleep} * T_{sleep} + P_{idle} * T_{idle}) \quad (3.1)$$

em que $P_{off}$, $P_{sleep}$ e $P_{idle}$ são os valores de potência em watts nos diferentes modos, enquanto $T_{off}$, $T_{sleep}$ e $T_{idle}$ são valores em percentagem do tempo anual em que o computador actuou em cada modo. O valor obtido $E_{TEC}$ será a energia consumida anualmente em unidades de kWh. Este valor baseia-se na tabela seguinte [54], onde se dão pesos percentuais aproximados do tempo de actividade dos diferentes modos numa base anual.

**Tabela 3.2** – Pesos percentuais aproximados do tempo de actividade dos diferentes modos numa base anual

| | ***Desktops*** |
|---|---|
| **Toff** | 55% |
| **Tsleep** | 5% |
| **Tidle** | 40% |

A tabela representada na imagem seguinte [54] apresenta então os valores TEC requeridos pela Versão 5.0 da especificação ENERGY STAR para computadores (neste caso apresentam-se só os requisitos para Computadores Pessoais e Computadores Pessoais Integrados).

| | **Desktops and Integrated Computers (kWh)** |
|---|---|
| **TEC (kWh)** | **Category A**: ≤ 148.0<br>**Category B**: ≤ 175.0<br>**Category C**: ≤ 209.0<br>**Category D**: ≤ 234.0 |
| | **Capability Adjustments** |
| Memory | 1 kWh (per GB over base)<br><br>*Base Memory*:<br>Categories A, B and C: 2 GB<br>Category D: 4 GB |
| Premium Graphics (*for Discrete GPUs with specified Frame Buffer Widths*) | Cat. A, B: 35 kWh (FB Width ≤ 128-bit)<br>50 kWh (FB Width > 128-bit)<br><br>Cat. C, D: 50 kWh (FB Width > 128-bit) |
| Additional Internal Storage | 25 kWh |

**Figura 3.2** – Valores de TEC especificados pela EPA [54]

### *Servidores*

Já para os servidores que normalmente integram os *datacenters*, como *Clusters* e *Grids*, estes são abrangidos pela Versão 1.0 (*Final Draft*) dos Requisitos do Programa ENERGY STAR para Servidores [55]. Tal como anteriormente, algumas definições complementares devem ser apresentadas para uma melhor compreensão da especificação [55].

1. Servidor: É uma máquina que providencia serviços e gere recursos de rede que são úteis

a outros dispositivos como Computadores Pessoais, telemóveis, PDAs, telefones IP, etc.. Os Servidores não são na maioria dos casos acessíveis ao utilizador por dispositivos de *input* tradicionais como o teclado ou o rato, mas são isso sim mais destinados a ser acedidos, receber e executar pedidos feitos normalmente através de uma ligação de rede. As máquinas devem ter as seguintes características para serem qualificadas como um Servidor segundo o programa ENERGY STAR:

- Publicitado e vendido como um Servidor;
- Concebidos e listados como suportando um Sistema Operativo (OS) para Servidores e/ou *hypervisors*, e destinados a executar aplicações empresariais instaladas pelos utilizadores;
- Suporte para Código Corrector de Erros (ECC) e/ou buffer de memória;
- Vendido com uma ou mais fonte de alimentação AC-DC ou DC-DC;
- Tem pelo menos um disco rígido instalado para armazenar e inicializar um OS local ou *hypervisor*;
- Todos os processadores têm acesso à memória partilhada e são independentemente visíveis para um único OS ou *hypervisor*.

2. Servidor Gerido (*Managed Server*): Servidor destinado a um elevado grau de disponibilidade num ambiente altamente gerido, como um *datacenter*. Um Servidor Gerido necessita de possuir todas as características seguintes:

- Capacidade de operar com fontes de alimentação redundantes; e
- Um controlador de gestão dedicado instalado (exemplo, processador de serviço).

3. Servidores *Dual-Node*: Um Servidor *Dual-Node* consiste em dois servidores ou nós que partilham uma ou mais fontes de alimentação e estão incorporados num único invólucro. A potência combinada de todos os nós é distribuída pela(s) fonte(s) de alimentação partilhada(s).

Os sistemas de servidores Blade, incluindo os Servidores Blade ou os Chassis Blade, não são elegíveis para qualificação segundo o programa ENERGY STAR. No entanto, como o *Cluster* **Ineb** da FEUP é um sistema deste tipo, apresenta-se aqui a definição destes sistemas.

4. Sistema Blade: É um sistema constituído por um Chassis Blade e um ou mais Servidores Blade ou uma ou mais unidades de armazenamento Blade. É um sistema concebido para ser uma solução escalável, permitindo que os seus operadores possam adicionar ou substituir facilmente tanto Servidores como unidades de armazenamento Blade no sistema total. Estes sistemas permitem então acomodar e operar eficientemente múltiplos Servidores ou unidades de armazenamento Blade num único invólucro.

5. Chassis Blade: Um invólucro que contém recursos partilhados - como fontes de alimentação para conversão de potência, armazenamento partilhado ou *hardware* para distribuição de potência DC, gestão de temperatura, gestão de sistema ou serviços de rede - para a operação de Servidores Blade ou unidades de armazenamento Blade. Um Chassis Blade compreende múltiplos níveis onde podem ser incorporadas unidades Blade de diferentes tipos.

6. Servidor Blade: Um Servidor que consiste, no mínimo, num processador e numa memória que se apoia em recursos partilhados (fonte de alimentação, refrigeração, etc) para operar.

Estes Servidores são destinados a ser instalados num Chassis Blade, já que são incapazes de operar independentes do Chassis, e são *hot-swappable*.

Assim, um Servidor deve ir de encontro à definição apresentada para ser elegível para qualificação ENERGY STAR. Como referido, os sistemas Blade não são considerados elegíveis nesta versão da especificação [55], devido em grande parte à maior dificuldade de se obter medições correctas da potência gasta por estes sistemas de servidores no estado *idle*, já que são sistemas mais complexos que por exemplo os sistemas de servidores em *rack*. A EPA espera numa versão seguinte desta especificação já incluir dados e requisitos de eficiência energética para estes sistemas. De notar ainda que a especificação também só abrange servidores com 4 ou menos processadores. Mais uma vez, é esperado a versão seguinte da especificação já expandir esta cobertura a servidores com mais de 4 processadores.

Pode-se ver então na tabela representada na imagem seguinte os requisitos de eficiência energética para a potência gasta pelos servidores em estado *idle*. Esta tabela é referente aos requisitos para servidores com uma ou duas *sockets* para processadores, ou seja, servidores que suportam no máximo dois processadores físicos. Um servidor que tenha 3 *sockets* para processadores mas só 1 processador instalado não estará sujeito aos requisitos desta tabela. As quantidades referidas na tabela são componentes efectivamente instalados no sistema e não o que o sistema suporta, ou seja, um sistema com 2 *sockets* mas apenas um processador instalado estará na categoria A ou B, consoante a definição a que corresponde.

| Computer Server System Type | Idle Power Limit |
|---|---|
| Category A: Standard Single Installed Processor (1P) Servers | 55.0 watts |
| Category B: Managed Single Installed Processor (1P) Servers | 65.0 watts |
| Category C: Standard Dual Installed Processor (2P) Servers | 100.0 watts |
| Category D: Managed Dual Installed Processor (2P) Servers | 150.0 watts |

**Figura 3.3** – Requisitos de potência para servidores em estado *idle* [55]

A figura anterior apresenta a tolerância de potência no estado *idle* para um sistema de Servidor básico, por exemplo, com mínima memória, uma única fonte de alimentação, um disco duro, uma configuração mínima de E/S. Servidores que correspondam à definição de Servidores Geridos apresentada anteriormente serão da Categoria B ou D, enquanto se não corresponderem a essa definição serão da categoria A ou C da tabela.

A tabela que se apresenta na figura 3.4 mostra as permissões de potência adicionais para componentes extra em relação ao sistema básico que os sistemas de servidores possam ter.

| System Characteristic | Applies To: | Additional Idle Power Allowance |
|---|---|---|
| Additional Power Supplies | Power supplies installed explicitly for power redundancy[1] | 20.0 watts per Power Supply |
| Additional Hard Drives | Installed hard drives greater than one | 8.0 watts per Hard Drive |
| Additional Memory | Installed memory greater than 4 GB[2] | 2.0 watts per GB[2] |
| Additional I/O Devices | Installed Devices greater than two onboard, 1 Gigabit (Gbit) Ethernet ports[3] | < 1Gbit[4]: No Allowance<br>= 1 Gbit[4]: 2.0 watts / Active Port[5]<br>> 1 Gbit[4] and < 10 Gbit[4]: 4.0 watts / Active Port[5]<br>≥ 10 Gbit[4]: 8.0 watts / Active Port[5] |

**Figura 3.4** – Tolerâncias de potência adicional nos servidores [55]

Os Servidores *Dual-Node* com 1 ou 2 *sockets* para processadores por nó devem seguir os valores de potência apresentados na figura 3.3 numa base por nó, desde que os nós do sistema tenham uma configuração e componentes idênticos entre si. No caso deste tipo de servidores, a potência por nó no estado *idle* seria encontrada medindo a potência total da unidade inteira e dividindo depois esse valor por dois. Por exemplo, para dois nós (Servidores) do sistema que partilhem a mesma fonte de alimentação, teria de se medir a potência no estado *idle* na fonte de alimentação, o que daria a potência total, sendo esse valor medido dividido depois por dois para achar a potência por nó. Esta potência por nó deverá então seguir os requisitos das tabelas apresentadas nas figuras 3.3 e 3.4.

***Servidores com mais de 2 sockets para processadores***

No caso dos Servidores com mais de 2 *sockets* para processadores, estes têm de permitir a gestão de potência a nível do processador para reduzir o gasto de potência em situações que não haja trabalho "útil" a ser efectuado pelo processador, como no caso do estado *idle*. Nesta especificação estes sistemas não têm limites estabelecidos como os previamente apresentados para sistemas de servidores com 1 ou 2 *sockets* para processadores. Estes sistemas têm de ser comercializados com esta funcionalidade de gestão de potência activa, e/ou um controlador de gestão ou processador de serviço. Se os sistemas já vierem com um OS ou um *hypervisor* previamente instalado, a funcionalidade mencionada também deverá constar por omissão nas funcionalidades do sistema.

Para ir de encontro a estes requisitos, todos os processadores devem, em períodos de baixa utilização, ser capazes de reduzir o consumo de potência através de:

- Redução da voltagem oferecida ao sistema e/ou da frequência através de *Dynamic Voltage* e *Frequency Scaling* (DVFS); ou
- Entrando num modo ou estado de baixo consumo energético do processador quando a respectiva *socket* não estiver em uso.

Para servidores *Dual-Node* com mais de 2 *sockets* por nó o mesmo requisito de gestão de potência é pedido.

### 3.3.2 – Temperatura e eficiência energética

O aumento do poder computacional traz consigo um novo problema para as entidades que possuam estruturas com um nível de computação elevado. A possibilidade de realizar tarefas que eram antes praticamente inatingíveis devido a este poder computacional leva a um aumento enorme nos custos de manutenção e operação de estruturas como *Clusters* ou *Grids*. Como visto anteriormente neste capítulo, os sistemas de refrigeração e arrefecimento dos *datacenters* estão a tornar-se numa das áreas destas estruturas que mais energia gasta, para fazer face à grande densidade de potência que plataformas cada vez mais capazes originam.

O custo de operação e manutenção ao longo da "vida" dos servidores e dos sistemas de refrigeração inerentes tem, hoje em dia, um custo semelhante ao da compra dos servidores e estrutura em si. [58] Assim, estes custos não podem ser ignorados e começam a ter um peso muito grande nos orçamentos das entidades que operam ou possuem *datacenters*, o que torna inevitável e vital encontrar medidas para os reduzir.

Com este intuito, o trabalho de Michael K. Patterson [58] estuda o impacto da temperatura em diferentes áreas da estrutura de um *datacenter* típico, para tentar aquilatar os possíveis ganhos de eficiência ao variar o *setpoint* de temperatura de um determinado *datacenter*. Este *setpoint* é o valor que os proprietários de determinado *datacenter* escolheram para ter como referência para a operação do *datacenter*, ou seja, se o seu centro de computação operar tipicamente a uma temperatura de 23ºC, será esse o *setpoint* de temperatura escolhido para o *datacenter*. De referir que esta temperatura se refere àquela medida na entrada do equipamento já que muito provavelmente a temperatura do espaço será superior a essa, principalmente nas salas onde há dissipação e descargas de calor e potência dos servidores, por exemplo.

Posto isto, existe um intervalo de valores para a temperatura dos *datacenters* que é aceite como parâmetro ou *standard* energéticos para estruturas de computação de elevado desempenho. Este parâmetro foi estabelecido pela Sociedade Americana de Engenheiros de Aquecimento, Refrigeração e Ar-Condicionado (*American Society of Heating, Refrigerating and Air-conditioning Engineers* - ASHRAE [51]) que recomenda que os *datacenters* mais críticos sejam mantidos numa temperatura entre os 20 e os 25ºC. Esta é considerada a temperatura óptima para uma boa performance computacional e boa manutenção dos componentes dos *datacenters*, permitindo um bom equilíbrio entre poder de computação e não degradação dos componentes, estimando-se que esta gama de temperaturas não force um grande aumento energético na operação das estruturas de arrefecimento inerentes a estes sistemas [58] [59]. Adicionalmente, a ASHRAE [51] [58][59] diz ainda que, tendo em conta a diversidade nas estruturas dos *datacenters*, já que não são todos iguais, por exemplo a nível de tamanho, localização geográfica, número de componentes, etc, existe uma gama mais vasta de temperaturas aceitáveis e permitidas para os *datacenters*. Assim, enquanto a gama recomendada é entre os 20 e 25ºC, a gama aceitável e permitida de temperaturas para estas estruturas é de 15 a 32ºC. Este é um parâmetro claramente aceite no cenário actual, já que a maioria dos *datacenters* opera dentro deste intervalo estabelecido pela ASHRAE [58].

No entanto, não fugindo à regra dos parâmetros e *standards* que começam a surgir para uma maior eficiência energética dos *datacenters*, mesmo dentro desta gama há alguma discussão sobre se o *setpoint* de temperatura deve ser baixo ou alto. Por exemplo, há quem

defenda que os *datacenters* devem operar com temperaturas baixas como 15 ou 16ºC, defendendo que mais frio é melhor, tanto para a performance das máquinas, como para a estabilidade do sistema [58]. Há também quem tenha a opinião contrária, afirmando que se pode poupar energia operando a temperaturas mais altas, por exemplo acima dos 25ºC até aos 32ºC do limite recomendado pela ASHRAE [58][59]. O estudo de Michael K. Patterson [58] tenta aquilatar das diferenças em eficiência energética nos *datacenters* em função da temperatura (nomeadamente dentro da gama recomendada pela ASHRAE).

Não é objectivo deste trabalho estudar este caso a fundo, mas será apresentado um breve resumo de algumas das conclusões que Patterson obteve no seu estudo [58] para mostrar que a temperatura pode efectivamente influenciar a eficiência energética dos *datacenters*. Neste estudo [58] Patterson tentou aquilatar do impacto da temperatura no gasto total de energia num *datacenter*. Para este fim Patterson [58] analisou as fontes de potência, incluindo a fonte ininterrupta de potência (UPS - *Uninterrupted Power Supply*), e o sistema de distribuição de potência; ainda foi objecto de estudo o impacto da temperatura nos próprios servidores e no uso de energia tanto nas unidades de fornecimento de potência dos servidores (PSU - *Power Supply Unit*) como nos CPUs, memórias e ventoinhas internas; finalmente, também considerou o gasto de energia eléctrica pelos sistemas de refrigeração, desde as unidades de tratamento do ar das salas de computação, as chamadas CRAHs (*Computer Room Air-Handling unit*) e os subsistemas de *chillers* e torres de refrigeração. Todas estas áreas contribuem para os gastos com electricidade e, portanto, energéticos do *datacenter*. Deste modo, Patterson [58] vai de encontro à ideia que defende, a de que um olhar para o gasto de energia como um todo no *datacenter* permite identificar as áreas e as alturas apropriadas para modular a temperatura das várias salas que constituem os *datacenters* de forma a ter ganhos na eficiência energética destas grandes infra-estruturas. Esta análise de cada componente contribuinte para o gasto energético do *datacenter* permite determinar a relação ***Potência = f(T)*** em cada componente, determinando a reacção de cada um a mudanças na temperatura a que operam. Assim verificou se a sua eficiência e consumo energético aumenta ou diminui com determinada alteração no *setpoint* de temperatura. De referir que Patterson [58] estuda os efeitos da variação da temperatura dentro dos limites estabelecidos pela ASHRAE [58][59] - 15ºC aos 32ºC.

De seguida apresenta-se o essencial das análises efectuadas no estudo de Patterson [58] a alguns dos componentes mais comuns dos *datacenters* e cujo consumo de potência é mais relacionado com a temperatura como os servidores e as suas ventoinhas internas, as CRAHs, as bombas refrigeradoras e os subsistemas de *chillers* e torres de refrigeração, e ainda as suas conclusões gerais sobre o impacto da temperatura no consumo energético destas infra-estruturas.

**Servidores:** Nos servidores há vários componentes que consomem energia, sendo os mais importantes e mais relacionados com a temperatura o CPU e as ventoinhas internas dos servidores. O uso de potência do CPU é bastante relacionado com a temperatura e devido a isso há já muita discussão na indústria que fabrica este tipo de peças sobre diferentes designs e implementações de CPUs que garantam menos *hot-spots* e por isso menos fugas induzidas termicamente. Os designs de CPUs mais antigos, ainda com tecnologias na ordem dos 0.35 a 0.18 micros, tinham menos fugas energéticas. Todavia, com o avanço tecnológico observado e a crescente redução das geometrias dos *chips* e do silicone que o constitui, os caminhos de fugas tornaram-se mais curtos e, consequentemente, a frequência e volume das fugas

aumentaram. Enquanto antes eram uma pequena parte do total, as fugas agora, devido aos factores anunciados, podem chegar a ser 50% do total de fugas energéticas no CPU. Por conseguinte, a meta dos fabricantes de CPUs é conseguirem ir reduzindo as geometrias limitando os *hot-spots*, para manter o nível de fugas abaixo dos 50% referidos. Vários autores, como Fallah [60] e Mukherjee [61] por exemplo, afirmam que as fugas energéticas referidas se relacionam directamente com a temperatura. No entanto, a gama de valores varia. Fallah [60] sustenta que por cada ºC de aumento de temperatura resultante da geração de vários processos de CPU, as fugas aumentam consistentemente em 2%. Por seu turno, Mukherjee [61] afirma que uma redução de 12ºC na temperatura dos *hot-spots* origina uma redução de 12% na potência das fugas. É natural aferir destes factos que com a diminuição crescente das geometrias dos componentes críticos dos CPUs, as taxas de fugas continuarão a variar com a temperatura a que operam os CPUs. Logo, uma mudança no *setpoint* de temperatura da sala pode afectar estas fugas energéticas dos CPUs dos servidores, factor que tem de ser considerado para uma decisão desse género.

**Ventoinhas internas dos servidores:** As ventoinhas internas dos servidores são outro componente em que a variação de potência que gastam está directamente relacionada com a temperatura. Quase todos os servidores modernos (como quase todos os portáteis modernos, por exemplo) têm ventoinhas com velocidades variáveis, que aumentam ou diminuem de intensidade consoante o arrefecimento que a máquina precisa. Em termos gerais, o algoritmo de controlo da velocidade da ventoinha (FSC - *Fan Speed Control*) do servidor mede várias temperaturas e ajusta então a velocidade da ventoinha de forma a que os vários componentes do servidor operem a uma temperatura nunca mais alta que a sua temperatura máxima de operação. O FSC tenta ainda que a ventoinha rode o mais devagar possível, indo de encontro aos limites acima estabelecidos, para que seja minimizado o seu barulho e potência operacional. É desta forma lógico que aumentando a temperatura, a ventoinha terá de girar mais rápido para garantir o arrefecimento apropriado do servidor, gastando assim mais potência para remover a quantidade requerida de calor residual. Patterson [58] afirma mesmo que a potência usada pelas ventoinhas internas dos servidores é o maior gasto de potência dependente da temperatura na plataforma do servidor, já que aumentar a velocidade da ventoinha significa um grande aumento na energia utilizada pela plataforma do servidor.

**Unidades de tratamento do ar das salas de computação (CRAHs) e bombas refrigeradoras:** Uma temperatura ambiente mais quente dá origem a ar mais quente a circular nas salas, que devido a ter uma densidade mais baixa do que se fosse mais frio é mais fácil de mover por estas unidades. Por conseguinte, pode-se concluir que um aumento de temperatura nas salas aumenta a eficiência energética destas unidades, embora esse mesmo ar quente tenha menos capacidade de refrigeração. No entanto, estas variações de eficiência são em pequena escala, o que faz com que o impacto geral tanto das CRAHs como das bombas refrigeradoras na eficiência das estruturas que compõem o *datacenter* seja reduzido.

**Chillers / Torres de refrigeração:** Para os servidores observou-se que uma temperatura mais alta acarretava consigo um maior desafio em termos térmicos para o design do CPU e para a sua operação e das ventoinhas internas. Para os subsistemas de *chillers* e torres de refrigeração, quanto maior for o diferencial de temperatura, ou seja, quanto maior a diferença

em relação à temperatura exterior, mais eficientes se tornam estes sistemas. Isto deve-se ao facto de ser mais fácil de rejeitar e dispersar o calor residual com uma maior diferença de temperatura. Por exemplo, o *chiller* COP pode ter um aumento de eficiência num intervalo de 1 a 2.5% aumentando a temperatura da água de refrigeração cerca de 1 grau Fahrenheit [62]. Consequentemente, aumentando a temperatura da sala, a temperatura da água de refrigeração aumentará igualmente. Os ganhos totais dependerão da percentagem da potência total que é representada pelo sistema de refrigeração.

Como se pode ver pela análise de Patterson, há componentes do *datacenter* cuja eficiência melhora com um determinado aumento de temperatura, como os *chillers*, e outros cuja eficiência aumenta com um decréscimo da temperatura ambiente a que operam, como os servidores e as suas ventoinhas internas. Portanto, a temperatura das salas dos *datacenters* é uma questão delicada e que não pode ser tomada de ânimo leve. Assumindo que uma sala de determinado *datacenter* opera dentro da gama de temperatura recomendadas pela ASHRAE (20ºC - 25ºC), por exemplo num *setpoint* de 23º C, seria bom testar os custos ou ganhos energéticos totais, ou seja, de todos os componentes do *datacenter*, desde servidores a sistemas de refrigeração e fontes de potência, etc, variando o *setpoint* de temperatura de forma pequena, como 2º C abaixo e acima dos 23º C.

### 3.3.3 – PUE e DCiE - Métricas Propostas pela Green Grid [6]

A Green Grid é uma organização não lucrativa composta por profissionais TI que procura ajudar e guiar os proprietários e utilizadores de *datacenters*, com base no estabelecimento de requisitos e boas práticas para a potência e refrigeração destas infra-estruturas, bem como de novas tecnologias que possam diminuir os custos energéticos das estruturas de computação de elevado desempenho. Esta organização acredita que várias métricas relevantes podem ser estabelecidas para ajudar os *datacenters* e seus proprietários a fazer melhores decisões sobre como melhorar as suas infra-estruturas ou sobre novas implementações de uma estrutura como um *datacenter*. Com este propósito, a Green Grid desenvolveu e apoiou o uso de duas métricas que idealmente poderão dar um contributo no sentido de constatar se o *datacenter* pode ser optimizado ou se será necessário construir um novo *datacenter* para aumentar a eficiência energética das estruturas [63]. Essas métricas são: o PUE - *Power Usage Effectiveness* - e o DCiE - *DataCenter infrastructure Efficiency*.

Num artigo anterior da Green Grid, estas duas métricas eram já apoiadas pela Green Grid como boas indicadoras da eficiência a nível do gasto de electricidade e potência dos *datacenters*. Desde aí o PUE foi amplamente aceite pela comunidade TI, mas o DCiE (na altura ainda DCE) teve uma taxa de sucesso reduzida, devido às dúvidas que existiam sobre o que era realmente a eficiência energética de um *datacenter*. No artigo de Belady, Rawson, Pfleuger e Cader [63], autores pertencentes à Green Grid, estes reafirmaram o conceito de PUE e redefiniram o DCiE para acabar com a confusão existente que advinha do conceito de DCE anterior, agora renomeado DCiE.

Assim, o PUE é definido como [63]:

PUE = Potência Total da Infra-estrutura / Potência do Equipamento TI (3.2)

O DCiE é o seu recíproco, ou seja [63]:

DCiE = 1 / PUE = Potência do Equipamento TI / Potência Total da Infra-Estrutura (3.3)

É útil definir nas duas equações o que é que cada termo compreende. Tem-se então que em (3.2) e (3.3) o termo Potência Total da Infra-estrutura se refere ao valor de potência que se obtém directamente a partir da medição efectuada no contador da infra-estrutura ou somando as potências que os diferentes componentes da infra-estrutura gastam, medidas em cada contador, se não houver um contador geral da infra-estrutura. A infra-estrutura referida é o *datacenter*, ou seja, interessa a potência que é dedicada a este apenas. Por exemplo, em edifícios que têm várias estruturas consumidoras de energia e que também albergam um *datacenter* nas suas instalações, só interessa a potência dedicada à operação deste. Nesta Potência Total da Infra-estrutura pode-se assim incluir tudo que suporta a carga do equipamento TI, como: Nós de computação, rede e armazenamento; componentes do sistema de refrigeração como *chillers*, torres de refrigeração, unidades de tratamento do ar e unidades de ar condicionado das salas de computação (CRAHs e CRACs respectivamente) e bombas refrigeradoras; componentes de distribuição de energia e fontes de energia como a fonte ininterrupta de potência (UPS), comutadores, unidades de distribuição de potência (PDU-*Power Distribution Unit*), baterias, geradores de potência e perdas de distribuição externas ao equipamento TI; outros consumidores de energia diferentes como o caso da iluminação do *datacenter* [63]. Já o termo Potência do Equipamento TI entende-se como todo o equipamento que permite a gestão, armazenamento, processamento e troca de informação e dados no *datacenter*. A Potência do Equipamento TI compreende então a potência gasta ao nível de equipamentos de computação, armazenamento e rede do *datacenter*. Ainda se pode incluir neste equipamento TI outros computadores (pessoais ou portáteis) usados para operar ou controlar de alguma forma as operações do *datacenter*, monitores ou *switches* KVM por exemplo.

O PUE e o DCiE permitem deste modo que os *datacenters* comparem a sua operação com a de outros *datacenters* e são bons indicadores para que os proprietários verifiquem se a sua infra-estrutura tem melhorado a nível de eficiência dos seus processos e do seu design com o passar do tempo e mesmo para determinar oportunidades de focar energia que é desperdiçada para equipamentos TI adicionais. Estas métricas oferecem assim a possibilidade de os proprietários de infra-estruturas de computação de elevado desempenho, como *Grids* operacionais ou *Clusters*, terem uma noção da alocação de energia na sua estrutura. Por exemplo, um PUE de 2.0 revela que o *datacenter* exige duas vezes mais energia que aquela necessária para operar os equipamentos TI. Imaginando que um servidor necessitava de 400 W para um funcionamento optimizado, o sistema (*Grid, Cluster*, etc) necessitaria de uma potência de 800 W para entregar esses 400 W ao servidor, se o sistema tivesse um PUE de 2.0. Adicionalmente, o DCiE associado seria de 50%, o que quereria dizer que os equipamentos TI desse *datacenter* representavam 50% do gasto energético deste último [63].

A figura seguinte mostra como o PUE e o DCiE seriam calculados num *datacenter*:

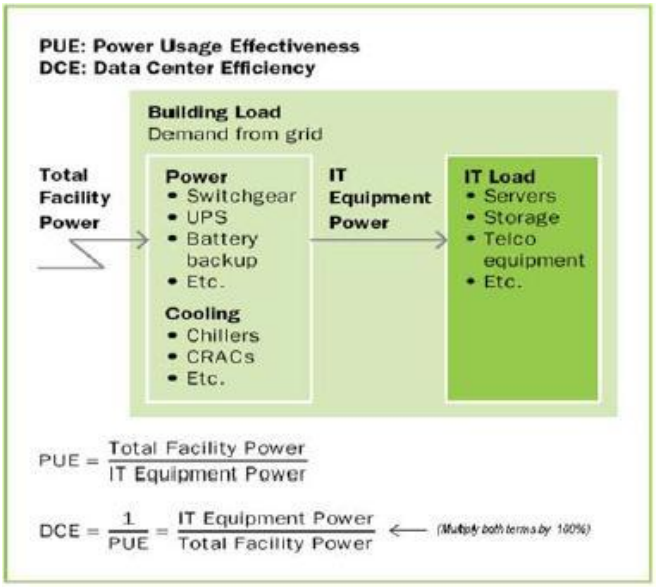



**Figura 3.5** – Ilustração de como o PUE e o DCiE seriam calculados num *datacenter* [63]

Na figura é apresentado o exemplo de um edifício com várias funções e estrutura e que também possui um *datacenter*. Verifica-se então que na medição do PUE se deverá medir a potência usada pelo *datacenter* apenas, sendo essa medição feita como referido anteriormente no contador geral do equipamento do *datacenter* ou nos contadores dos diferentes componentes constituintes do sistema, já previamente enumerados e representados na figura antes da carga para os equipamentos TI. Ainda de referir que o ponto mais provável de medição da potência usada pelos equipamentos TI será nos *outputs* das unidades de distribuição de potência das salas de computação, onde se encontram por exemplo as *racks* de servidores [63].

O PUE pode ir teoricamente de 1.0 até ao infinito. Não há ainda grandes medições ou conjuntos de valores estatísticos que permitam estabelecer uma gama mais ou menos precisa de valores médios do PUE para variados *datacenters*. No entanto, estudos preliminares indicam que uma grande parte dos *datacenters* têm um PUE de 3.0 ou maior, mas que com um design apropriado ou com a implementação de medidas e estratégias que visem a melhoria de eficiência nos *datacenters* seja possível alcançar valores do PUE até perto do 1.6 [63] [64]. De referir que um PUE de 1.0 significaria 100% de eficiência, sendo esse o *datacenter* ideal, em que toda a potência seria usada apenas pelos equipamentos TI. É possível ver pela tabela apresentada na imagem seguinte [65] o significado dos diferentes valores do PUE e DCiE:

| PUE | DCiE | Level of Efficiency |
|---|---|---|
| 3.0 | 33% | Very Inefficient |
| 2.5 | 40% | Inefficient |
| 2.0 | 50% | Average |
| 1.5 | 67% | Efficient |
| 1.2 | 83% | Very Efficient |

**Figura 3.6** – Significado em eficiência dos diferentes valores de PUE e DCiE [65]

A Green Grid sugere que os *datacenters* comecem a usar estas métricas o quanto antes para poderem gerir mais eficazmente as suas operações e a sua evolução, de forma a reduzir custos energéticos e monetários e a minimizar a sua pegada ecológica. Embora seja ainda difícil fazer estas medições, devido ao intrincado caminho de PDUs, transformadores, geradores, *chillers* e outros componentes, sugere-se que estas não sejam efectuadas apenas uma vez mas sim em intervalos regulares e seguidos ao longo do tempo [63] [65]. Este cálculo regular do PUE e DCiE permite por exemplo obter metas de eficiência e determinar onde e quando há oportunidades de poupar energia, como no exemplo a seguir [65].

| Basic Calculation | |
|---|---|
| Total IT Load | 94kW |
| Total Facility Load | 200kW |
| **PUE** | **2.13** |
| **DCiE** | **47%** |

**Figura 3.7** – Exemplo do cálculo básico do PUE e do DCiE [65]

Aqui vê-se que dos 200 kW usados pela infra-estrutura total, apenas 94 kW são usados pelo equipamento TI, o que dá um PUE de 2.13 e um DCiE de 47%. Há assim uma diferença de 106 kW, valor de energia que não é efectivamente usada pelos equipamentos que realmente movimentam, armazenam e processam os dados no *datacenter*. Se for feita uma análise ainda mais detalhada, poderia obter-se as seguintes informações:

| Detailed Calculation | |
|---|---|
| Total IT Load | 94kW |
| Cooling Infrastructure | 80kW |
| Power System Load | 24kW |
| Lighting Load | 2kW |
| Total Facility Load | 200kW |
| **PUE** | **2.13** |
| **DCiE** | **47%** |

**Figura 3.8** – Exemplo do cálculo detalhado do PUE e do DCiE [65]

Por aqui se vê que 80 kW dos 200 kW são só para o sistema de refrigeração implementado e 2 kW são gastos em iluminação. Com esta análise detalhada, os proprietários destas infra-

estruturas poderão determinar áreas críticas do *datacenter* nas quais podem actuar e poupar energia sem prejuízo da performance do seu sistema.

### 3.3.4 – DCeP

O debate sobre as melhores formas de implementar e medir a evolução na eficiência energética de estruturas de computação de elevado desempenho tem vindo a ganhar força na comunidade TI nos dias de hoje. Enquanto o PUE e o DCiE começam a ser amplamente aceites por esta comunidade, estas métricas focam-se mais na eficiência das estruturas físicas, como os sistemas mecânicos e eléctricos que suportam os *datacenters*, não indo de encontro ao problema da eficiência aliada à produtividade nos *datacenters* [66]. Nesse sentido, e também com o apoio da Green Grid, uma nova métrica tem surgido e ganho força como podendo dar uma resposta a este problema. Esta métrica é o DCeP - *DataCenter energy Productivity*.

O DCeP é uma métrica que quantifica o "trabalho útil" realizado pelo equipamento TI de um *datacenter* comparado com a energia que esse "trabalho útil" requer para ser feito. É uma forma de tentar determinar a produtividade do *datacenter*. Portanto, o DCeP é definido como [66]:

DCeP = Trabalho Útil Produzido / Total de Energia Consumida pelo *Datacenter* ao longo do tempo (3.4)

Tal como o PUE e o DCiE, o DCeP também deve ser calculado em intervalos regulares para verificar a sua evolução ao longo do tempo. O tempo é um factor importante para esta métrica. O DCeP tem uma "janela de avaliação" em que o "trabalho útil" e a energia são comparados em relação a um prazo definido pelo utilizador [66]. Pode ser obtido para um equipamento TI singular ou por exemplo para um aglomerado (*Cluster*) de equipamentos de computação.

O termo "Trabalho Útil" apresentado na equação (3.4) pode ser compreendido como sendo dependente de duas leituras, as tarefas realizadas pelo *hardware*, que devem ser o mais específicas possível e a "janela de avaliação". De acordo com a Green Grid, esta "não deve ser inferior a cerca de 20 vezes a média de tempo de execução de qualquer uma das tarefas iniciadas na "janela de avaliação"" [66]. Cada *datacenter* deve levar em conta o seu modelo de design e operação bem como a sua carga de trabalho típica aquando do cálculo destes dois valores. A seguinte equação é proposta pela Green Grid para calcular o "trabalho útil" [66]:

Trabalho Útil = (STT * VT) * FUBT * TAR (3.5)

em que:
STT = Somatório de todas as tarefas
VT = Valor da tarefa
FUBT = Função de utilidade baseada no tempo
TAR = Tempo absoluto de realização

Pode-se verificar a partir da equação (3.5) que a Green Grid faz a distinção entre tarefas, pois é sabido que há tarefas mais importantes que outras, podendo ser tarefas críticas ou não. Para simplificar o cálculo, é possível assumir que as tarefas têm todas o mesmo valor, sendo assim tanto a parcela VT como a parcela FUBT igual a '1', vindo assim [66]:

Trabalho Útil = STT * TAR (3.6)

Neste caso, o "trabalho útil" seria então o número de tarefas (trabalhos, transmissão de dados, etc) que o *hardware* realiza durante o tempo estabelecido pela "janela de avaliação".

Já o Total de Energia Consumida pelo *Datacenter* ao longo do tempo envolve dois valores distintos: a energia eléctrica efectivamente gasta pelo *hardware* que executa as várias tarefas de computação da infra-estrutura, em kWh; o PUE do *datacenter* [66]. Então, vem:

Total de Energia Consumida pelo *Datacenter* ao longo do tempo = kWh de energia consumida pelo *hardware* * PUE do *datacenter* (3.7)

Mais uma vez, poderá não ser fácil efectuar este tipo de medições, já que os *datacenters* terão de possuir instrumentos de medição apropriados para cada componente do sistema e terem implementados programas e meios que lhes permitam medir o PUE da infra-estrutura. Muitos *datacenters* ainda não possuem este tipo de capacidades, mas as novas tecnologias que vêm surgindo ao nível de contadores de energia, dispositivos que medem a potência nos *racks* de *Clusters* e novo *software* fazem com que a implementação de uma métrica deste tipo seja possível nos tempos próximos [66].

Concluindo, uma medição regular do PUE e DCiE das infra-estruturas permite aos proprietários aperfeiçoar ou remediar áreas críticas do seu sistema e melhorar a eficiência energética deste. Estes pequenos progressos na estrutura física devem ser o ponto de partida para os proprietários e utilizadores dos *datacenters*. De seguida, poderão concentrar-se em obter medições consistentes da produtividade dos seus sistemas de computação, para encontrar novas formas de a melhorar. Este processo pode trazer uma nova visão sobre áreas e alturas oportunas de modernizar, consolidar, virtualizar ou mesmo desmantelar plataformas de computação mais velhas e praticamente inactivas ou inúteis. O cálculo do PUE, DCiE e DCeP permite fazer um correcto (re)dimensionamento das infra-estruturas de computação de elevado desempenho que permita que a performance destas seja optimizada e vá de encontro aos requisitos da sua linha de negócio e maximizar lucros, minimizando a sua pegada ecológica.

### 3.3.5 – CPE

No seguimento dos parâmetros energéticos estabelecidos pelo PUE, DCiE e DCeP, surge também o CPE - *Compute Power Efficiency*. Esta métrica é, tal como o DCeP, prometedora mas necessita ainda de mais desenvolvimento e discussão antes de poder ser um parâmetro corrente para os proprietários de estruturas TI terem em conta. O CPE poderá ajudar os

proprietários dos *datacenters* a optimizarem o seu sistema, de forma a que uma maior eficiência não só energética mas também computacional seja atingida [67].

O CPE define-se como [67]:

$$CPE = UET * PET / PTI = UET / PUE \quad (3.8)$$

em que:
UET = Utilização do Equipamento TI
PET = Potência do Equipamento TI
PTI = Potência Total da Infra-estrutura

Como se vê pela fórmula do CPE (3.8), esta métrica providencia uma ideia da qual a percentagem da potência total fornecida à infra-estrutura é realmente usada para tarefas de computação. Liga assim a eficiência de toda a estrutura do *datacenter* com a eficiência do seu equipamento TI permitindo aquilatar quão eficientemente a potência do *datacenter* é usada para computação [67].

O CPE pode ser calculado para um servidor individual ou para o conjunto de servidores do *datacenter*, desde que uma média ponderada da utilização da estrutura de servidores total seja de obtenção viável. Em muitos *datacenters*, a utilização de servidores de uma empresa é tipicamente de 20% ou menos, o que quer dizer que para um *datacenter* típico com um PUE de 2.0, o CPE será 10%. Assim, 1W de cada 10W de potência total no *datacenter* é usado para tarefas de computação. Estudos preliminares do CPE indicam que a maioria dos *datacenters* desperdiça energia inutilmente para fornecer potência a estruturas de computação e sistemas de refrigeração, havendo um provisionamento bastante acima do requerido para essas estruturas [67][68].

### 3.3.6 – Outros parâmetros energéticos relevantes nos *datacenters*

Além dos parâmetros, *standards* e métricas apresentados até agora, existem outros parâmetros energéticos ou métricas que são importantes de seguir num *datacenter*. Estes são parâmetros que não têm valores por referência ou gamas recomendadas de valores estabelecidas, como visto para a temperatura ou potência, mas que podem fornecer informações muito relevantes sobre o gasto energético dos *datacenters*, a sua carga de trabalho ou simplesmente para a sua monitorização, observando qualquer comportamento fora do normal de qualquer componente. Estes são parâmetros normalmente monitorizados pelas plataformas de monitorização utilizadas ou mesmo, como se mostrou anteriormente neste trabalho, pelo escalonador usado nos *Clusters* ou *Grids*. De referir que para uma melhor monitorização estes parâmetros devem ser consistentemente monitorizados, actualizando o seu valor em intervalos de tempo regulares, quanto mais pequenos possíveis melhor. Por conseguinte, tem-se:

**Velocidade das Ventoinhas dos Servidores:** Este parâmetro mede as rotações por minuto das ventoinhas internas dos servidores. Estas ventoinhas servem para arrefecer a temperatura de vários componentes dos servidores e a monitorização das suas rotações por

minuto poderá indicar anomalias nas máquinas, por exemplo de sujidade a mais que faz com que o servidor trabalhe em esforço e aqueça demasiado, carga de trabalho muito grande para o servidor ou mesmo defeito de algum componente interno do servidor, casos que poderiam fazer as ventoinhas aumentar bastante as rotações por minuto, o que indicaria uma anomalia na máquina.

**Pacotes Enviados / Recebidos:** Esta métrica indica quantos pacotes de dados são enviados ou recebidos por cada servidor ou nó do *Cluster* ou *Grid*. É importante para verificar qual a carga de trabalho que o servidor está a executar ou executou no momento, podendo ser importante para efeitos estatísticos da utilização de determinado nó ou *Cluster*, por exemplo.

**Espaço Total no Disco / Espaço Disponível no Disco:** Estas são duas métricas bastante relevantes. Cada servidor tem um espaço de disco total que não deverá ser ocupado totalmente para não condicionar a performance das tarefas realizadas pelo servidor. Assim, ao analisar qual a quantidade de espaço ainda livre no disco poderá por exemplo conseguir-se uma ideia sobre se será necessário aumentar o espaço no disco ou não ou se a carga de trabalho associada ao servidor é excessiva.

**Contagem de CPUs:** É muito comum por razões de licenciamento que seja necessário saber exactamente quantos CPUs físicos existem num servidor. Esta métrica ajuda a essa percepção.

**Velocidade do CPU:** Esta métrica mede então a velocidade a que o CPU está a operar. É importante no sentido de verificar se há quebras grandes na velocidade de operação do CPU do servidor, o que pode sugerir anomalias operacionais ou de *hardware* e perda de produtividade do mesmo. Embora isoladamente não seja um indicador 100% fiável da performance do servidor, é um factor importante para esse aspecto.

**CPU User:** Percentagem de CPU que é consumida por aplicações que o utilizador faz ou mandou fazer no servidor.

**CPU Nice:** Percentagem de CPU que é consumida ao correr múltiplos processos que requerem mais ciclos, ou seja, maior frequência, que o CPU pode providenciar.

**CPU System:** Percentagem de CPU que é consumida por aplicações que o sistema operativo impõe ao servidor, ou seja, por chamadas do sistema.

**CPU WaitIO:** Percentagem de CPU que está à espera que processos de *Input* ou *Output* do disco sejam terminadas.

**CPU Idle:** Percentagem de CPU que não está a fazer "trabalho útil" nenhum, que não corre nenhum processo ou aplicação. Esta é uma métrica muito importante no sentido do *Green Computing*, na medida em que os CPUs de muitos dos nós em muitos *datacenters* se encontram na maioria das vezes com uma elevada percentagem neste estado, portanto desperdiçando energia sem fazer qualquer "trabalho útil" de computação.

**Média da carga de trabalho do servidor:** Parâmetro que revela qual a média da carga de trabalho no servidor. Pode ser importante para verificar se há uma carga muito elevada num servidor comparando com outros servidores, por exemplo, o que pode indicar uma utilização excessiva desse nó.

**Swap Livre:** Indica qual a quantidade de área de *swap* livre no servidor. A área de *swap* é uma área do disco que permite guardar temporariamente uma imagem de um determinado processo. Quando a memória livre no servidor for suficiente as imagens guardads na área de *swap* são trazidas para a memória física novamente. Ter espaço livre suficiente na área de *swap* permite que o servidor tenha sempre alguma memória física livre, o que optimiza a performance do sistema. Esta é também uma métrica importante.

**Memória Total:** Memória total do servidor.

**Memória Livre:** Métrica que indica qual o espaço de memória livre existente no servidor. Quanto mais espaço de memória melhor a performance do sistema, sendo esta uma métrica importante para alertas quando se ultrapassa um valor limite de pouca memória no sistema, por exemplo.

**Total de Processos:** Métrica que indica o total de processos que o servidor tem no momento, a correr ou não.

**Total de Processos a Correr:** Métrica que indica o total de processos que o servidor está a executar no momento.

**Bytes In / Out:** Métrica que indica o número de bytes que estão a entrar ou a sair do sistema, nomeadamente por processos de rede como transferências de pacotes de dados.

Em jeito de conclusão, estes são parâmetros importantes de monitorização em sistemas como *Clusters* ou *Grids*, pois a análise conjunta de várias destas métricas permite analisar e qualificar a performance do sistema, e mesmo coleccionar dados estatísticos importantes sobre a operação de cada nó de um *Cluster* ou *Grid*.

Posto isto, uma matriz ilustrativa dos vários parâmetros e métricas existentes e mais relevantes para infra-estruturas de computação de elevado desempenho como o são os *datacenters* foi efectuada e pode ser consultada no apêndice A.

# Capítulo 4

# Solução Proposta

## 4.1 – Introdução

A realização de uma metodologia que permita que os consumos energéticos nos *datacenters* baixem e, consequentemente, também sejam reduzidos quer os gastos monetários quer a degradação dos materiais e performance dos sistemas de *Cloud Computing*, *Grids* ou *Clusters* é muito importante para os proprietários de estruturas deste tipo. De seguida, apresenta-se a metodologia que foi idealizada para o problema enunciado neste projecto.

## 4.2 – Metodologia desenvolvida

A solução que será apresentada foi idealizada tendo em conta o estudo anteriormente feito sobre as tecnologias e estruturas que geralmente são constituintes dos grandes centros de computação de elevada performance e, em particular, de acordo com o que existe na FEUP. Esta também deverá levar em conta o estudo sobre soluções de *Green Computing* estudadas e os parâmetros energéticos encontrados para estruturas deste género.

A solução idealizada considerou a estrutura existente na FEUP. A metodologia de solução a implementar deveria relacionar-se com o escalonador da estrutura GridFEUP e, se possível, relacionar-se com alguns dos parâmetros energéticos estudados.

O conceito de solução encontrado foi então a elaboração de um programa computacional que fizesse a ligação ao escalonador da infra-estrutura GridFEUP e dos seus *Clusters* e cujo funcionamento permitisse um corte nos gastos energéticos da estrutura mencionada. Como referido anteriormente neste trabalho, a FEUP possui três *Clusters* de elevado desempenho

que são geridos pelo CICA. Estes apresentam uma arquitectura do tipo *Master* / *Slaves*.

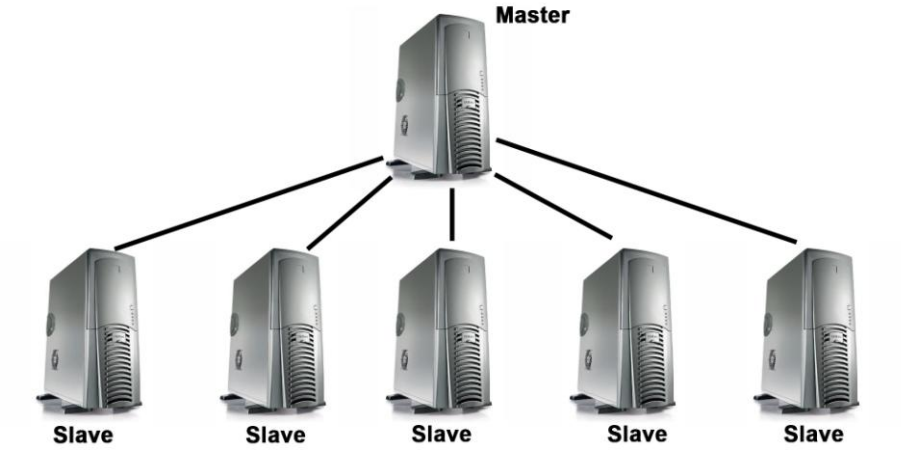



**Figura 4.1** – Arquitectura de cada Cluster, do tipo Master / Slave

O escalonador está presente no Master de cada *Cluster* e portanto é este nó que faz a gestão de todos os restantes nós desse mesmo *Cluster*. Ou seja, é a partir deste nó que saem as ordens do escalonador para os nós *Slaves*, os chamados *working nodes*, pois são estes que fazem o trabalho de computação propriamente dito. Estas ordens que partem do nó *Master* podem ser muito diversas, como submeter trabalhos para os *working nodes*, ligá-los ou desligá-los, retirar-lhes trabalhos, observar estatísticas dos nós ou do *Cluster* geral, mostrar o estado dos recursos, etc. É possível também ver se os nós do *Cluster* ou *Grid* estão em estado *busy* ou *running* (a executar "trabalho útil"), em estado *down* (desligados) ou em estado *idle* (ligados mas sem executar nenhum trabalho).

Devido a esta gestão centralizada de cada *Cluster* pensou-se que o programa a realizar se associasse ao escalonador, pois seria dessa forma mais fácil actuar sobre os nós, já que as acções originadas pela execução do algoritmo a realizar partiriam do nó *Master* para os *Slaves*. Deste modo também permite que a solução seja escalável, um factor bastante importante em sistemas deste tipo. Como se pode inferir dos vários estudos feitos e aduzidos neste trabalho, os *datacenters*, e particularizando no caso da FEUP, os seus *Clusters*, podem sempre ser objecto de modernizações ou melhorias nas suas infra-estruturas físicas. Uma solução escalável seria então essencial, para fazer frente a essas possíveis alterações. Por exemplo, se o algoritmo fosse desenvolvido para correr em cada nó do *Cluster*, sempre que se aumentasse o número de nós teria de ser instalado nesse novo nó e possivelmente modificado. Sendo executado no nó *Master*, as mudanças a efectuar no algoritmo se ocorressem algumas das mudanças mencionadas seriam mínimas e muito rápidas de fazer, sendo que não seria preciso sequer uma nova instalação do algoritmo nos novos nós ou mudanças muito grandes no caso de uma nova plataforma de servidores. Assim, a decisão de ligar o algoritmo ao escalonador do nó *Master* foi prontamente tomada de início e a metodologia idealizada deveria ter isso em conta.

Por conseguinte, com todos estes factores em mente, o algoritmo foi pensado da recorrendo às estatísticas de uso dos *Clusters* da infra-estrutura GridFEUP, deveria ser criado um histórico de utilização dos recursos de cada *Cluster* ao longo de vários anos, com valores médios diários, para ver quantos nós eram usados por dia em cada *Cluster*. Esse valor do

histórico seria uma variável, ou seja, havendo um valor de histórico por dia, no início de cada dia o valor do histórico para esse dia iria ser lido e armazenado numa variável. Como o programa se relaciona com o escalonador, após ler o valor do histórico este deveria aceder à fila de trabalhos do *Cluster*, a chamada *queue*. Se não houvesse trabalhos em fila de espera, seriam mantidos ligados apenas o número de nós correspondentes ao histórico, desligando os restantes nós que estivessem em estado *idle* e que faziam com que esse número estatístico diário fosse excedido. Se houvesse trabalhos em fila de espera, nada aconteceria devido ao valor estatístico do histórico.

Depois disto ocorrer, o programa deveria executar actualizações do estado do *Cluster* a cada minuto no restante tempo do dia. Ou seja, a cada minuto a fila de trabalhos no *Cluster* deveria ser verificada, havendo decisões a tomar consoante o valor da fila de trabalhos. Se a fila fosse diferente de zero, não se alterava nada em termos de nós, ficando o mesmo número de nós ligado que já se encontrava nesse instante. Posto isto, se no minuto seguinte a fila tivesse crescido, ou seja, se o seu valor fosse maior, seria ligado mais uma máquina que estivesse em *down*, pois poderia indicar que as máquinas ligadas não eram suficientes para executar os trabalhos em fila de espera, daí eles não terem sido aceites. Se a fila baixasse de valor o número de nós ligados não se alteraria pois indicaria que esses nós estavam a conseguir executar os trabalhos na fila e esta ia diminuindo. Se não houvesse nenhum trabalho em fila de espera, metade dos nós *idle* do *Cluster* seriam desligados pois não havia razão para estarem ligados desperdiçando energia. Também foi pensado que quando se refere o facto de desligar nós, seria mais aconselhável em vez de os desligar de vez pô-los em *stand-by* ou hibernação. Desta forma, evitava-se que houvesse ciclos de *shutdown* e *boot* consecutivos, o que gastaria mais energia do que deixando o nó ligado e poderia haver perda de performance do sistema. Ou seja, imaginando que um nó era desligado completamente (ciclo *shutdown*) num instante mas que era necessário no minuto seguinte, este teria de voltar a ser ligado. Para isso teria de esperar o tempo de *shutdown* mais o tempo do ciclo *boot* para o ligar, o que poderia prejudicar a performance do sistema. Depois, como já referido previamente neste trabalho, os ciclos de *boot* e *shutdown* são alturas em que as máquinas gastam também bastante potência, portanto uma situação destas gastaria um valor elevado de potência desnecessariamente. Se as máquinas fossem postas em hibernação, demoravam muito menos a entrar nesse estado, gastando menos potência para o fazer, e se fosse necessário um ou dois minutos depois regressariam ao activo muito rapidamente.

Adicionalmente a este conceito de solução, pensou-se ainda em incorporar duas variáveis de forma a optimizar o algoritmo. Estas variáveis são os parâmetros energéticos temperatura e potência, que são dois dos parâmetros mais importantes relacionados com a eficiência energética e que têm valores por referência que se recomenda que sejam cumpridos para uma boa eficiência. Assim, na altura de desligar os nós em *idle*, em vez de serem desligados metade aleatoriamente, escolhiam-se os melhores nós a desligar do conjunto de nós *idle*. Desse conjunto seria observado a temperatura e potência, escolhendo os nós *idle* que apresentassem valores mais elevados destas duas variáveis, sendo esses os nós desligados.

Este conceito de solução parece bastante viável, pois as *Grids* raramente têm de operar a 100% da sua capacidade, sendo que a estrutura GridFEUP não é excepção. Com esta solução, evitar-se-ia o desperdício energético que é ter muitos nós que estão em *idle* ligados por tempos excessivos, podendo passar muitos dias, semanas ou até períodos de tempo maiores em que esses nós estão ligados inutilmente. Esse facto faz aumentar o gasto

energético destas estruturas e a sua "pegada" ecológica, não aumentando em nada a sua performance. Ainda para mais, com mais servidores ligados, mais potência é gasta, logo mais calor residual é desperdiçado, aumentando assim também o gasto das estruturas que fazem a refrigeração do sistema. Também desta forma aumenta a temperatura das salas de computação, o que como vimos pode ser bastante prejudicial para a eficiência energética de alguns componentes e mesmo a sua performance, sendo ainda este um factor que pode levar à deterioração do *hardware* das salas de computação. Portanto, a solução a implementar é teoricamente um bom ponto de partida para aumentar a eficiência energética da estrutura GridFEUP, neste caso. Outro factor positivo desta solução é que não implica mudanças de fundo a nível da estrutura física do sistema, permitindo que não haja quebras de performance para os utilizadores aquando da sua implementação. Aliado a este facto, consequentemente esta é uma solução praticamente sem custos associados.

## 4.3 – Algoritmo e sua Implementação

### 4.3.1 – Dificuldades e Desenvolvimento

Com o conceito da solução idealizado avançou-se para a implementação prática dessa metodologia. Tendo em conta que se deveria relacionar com o escalonador dos *Clusters* da FEUP, neste caso o Moab já previamente estudado neste trabalho, foi necessário verificar como relacionar o *script* com o escalonador. Após o estudo do Moab verificou-se que este permite que *scripts* externos ao programa corram como se fossem parte do *software* e façam uso de comandos do Moab. Neste caso, a linguagem que é recomendada para estes *scripts* é o Perl. Esta foi a primeira dificuldade, pois o Perl era uma linguagem desconhecida e não utilizada por nós anteriormente. Por isso, um estudo desta linguagem teve de ser efectuado para analisar qual a sua forma de programação e capacidades.

Terminado o estudo e teste das características e capacidades do Perl, a tarefa seguinte foi obter o histórico de utilização dos *Clusters*. Infelizmente, surgiu uma contrariedade neste caso. Após alguma pesquisa dos dados, concluiu-se que as únicas estatísticas de utilização viáveis de serem usadas para construção do histórico eram mensais. Ainda assim, não havendo dados estatísticos diários de utilização dos nós da estrutura já tratados e prontos a serem usados, era possível obter dados sobre a utilização diária dos nós para um período de uma semana ou mesmo obter estatísticas de hora a hora para um período de um ou dois dias, através do portal do Moab que gere a infra-estrutura GridFEUP. No entanto, o nível de processamento de dados e informação necessários para gerar uma estatística diária de utilização dos *Clusters* para um período de um ano era incomportável para os computadores da FEUP que o estavam a fazer. Ainda se tentou fazê-lo mas o processo não gerou resultados nenhuns durante um grande período de tempo que se fez o teste, não sendo viável a obtenção destes dados. Assim, as estatísticas usadas para a elaboração do histórico são mensais, o que por sua vez introduz um factor de erro maior na estimativa da média de nós usados efectivamente nos *Clusters* da FEUP. O cálculo aproximado da média mensal de nós utilizado por *Cluster* na FEUP é apresentado no Apêndice B. Como consequência para o algoritmo idealizado, foi alterado a verificação do histórico de dia a dia para mês a mês. Assim, às 10 horas do primeiro dia de cada mês o histórico para cada *Cluster* é verificado. Um exemplo

deste processo são as seguintes linhas do programa implementado:

```
my @date = localtime(time);
if ($date[3] == 1 && $date[2] == 10 && $date[1] == 0) {
  $HistoricoGrid = &getHistoryGrid();
```

Aqui observa-se o uso de uma função nativa do Perl, a função 'localtime(time)'. Esta retorna um array de elementos em que cada posição tem um significado, da forma seguinte:

- [0] Segundos;
- [1] Minutos;
- [2] Hora;
- [3] Dia do Mês;
- [4] Mês (retorna em valor, com Janeiro = 0, Fevereiro = 1, etc);
- [5] Ano
- [6] Dia da Semana (retorna em valor, com Domingo = 0, Segunda-Feira = 1, etc);
- [7] Dia do Ano;
- [8] Quando este elemento é 1, indica horário de Verão.

O valor do histórico é então verificado ao primeiro dia de cada mês. Este é um exemplo de uma função, neste caso a 'getHistoryGrid()', que vai ler o valor do Histórico para o *Cluster* IBM/GridFEUP da FEUP:

```
sub getHistoryGrid() {
  my $line;
  my $valor;
  my @valores;
  my $FH = new FileHandle;
  my @date = localtime(time);

  $FH -> open("<HistoricoGrid.txt") or die "HistoricoGrid.in is Missing!";
  foreach $line(<$FH>) {
      @valores = split(' ',$line);
      if ($date[4] == $valores[0]) {
          $valor = $valores[1];
      }
  }
  close($FH);
  return $valor;
```

Pode-se verificar pela análise do código da função que é aberto o ficheiro HistoricoGrid.txt para leitura. O conteúdo deste ficheiro é o seguinte:

```
0 9
```

```
1 11
2 9
3 9
4 11
5 9
6 7
7 7
8 11
9 9
10 11
11 11
```

Ou seja, a primeira coluna do ficheiro vai servir para identificar o mês, já que a função 'localtime(time)' retorna o valor do mês como um escalar. A segunda coluna será o valor da média mensal de nós utilizados no *Cluster* IBM/GridFEUP. Para os outros dois *Clusters* da FEUP, o Idmec e o Ineb, a estrutura do ficheiro do histórico de utilização será igual.

Como também se pode observar pela análise da função anteriormente apresentada, esta é para obter o histórico do *Cluster* IBM/GridFEUP. No código total do algoritmo há outras funções que fazem o mesmo mas para os *Clusters* Idmec e Ineb. Esta necessidade de fazer esta divisão por *Cluster* no *script* implementado resulta de outra vicissitude encontrada no desenvolvimento do protótipo a implementar que foi o facto de num primeiro momento o protótipo ter sido desenvolvido e testado para um sistema de teste que não o sistema real da FEUP. Este sistema de teste tinha a arquitectura seguinte:


**Figura 4.2** – Arquitectura do sistema de teste

Como se pode verificar pela figura anterior, este sistema tinha apenas dois computadores, um computador que era o *Master* e um *Slave* ou *working node* gerido pelo *Master*. Este sistema de teste pretendia simular um *Cluster* e o seu funcionamento, sendo claro numa escala muito mais reduzida e não com servidores, mas sim constituído por dois computadores normais Hp Compaq dc7600 a simular dois servidores. Conseguiu-se sensibilizar o *staff* e gestores do Moab a garantirem uma licença temporária do Moab, neste caso a última versão deste *software* de escalonamento que já incorpora capacidades de *Green Computing*. O primeiro protótipo era então para este sistema, muito mais pequeno do que um *Cluster* real e

claro sem a carga de trabalho que a estrutura GridFEUP apresenta. Enquanto este sistema de teste dava para testar as primeiras tentativas de implementação do protótipo, permitindo aprimorar o código, optimizá-lo para tornar a sua execução mais rápida e ir corrigindo erros que foram aparecendo no seu desenvolvimento, este sistema era ainda muito redutor comparando com a infra-estrutura GridFEUP e não era suficiente para testar a total extensão que o protótipo deveria ter se se quisesse implementá-lo no sistema real. Por exemplo, só existindo um *working node*, não havia nós com diferentes estados mas sim sempre o mesmo nó sempre com o mesmo estado. As figuras 4.3 e 4.4 revelam a descrição do *working node* e da fila de trabalhos do sistema de teste, informações obtidas através cós comandos 'mdiag –n' e 'showq' do Moab.

```
[ee03268@gridssh ~]$ ssh root@moab01.fe.up.pt
root@moab01.fe.up.pt's password:
Last login: Sat Jun 26 05:45:32 2010 from gridssh.fe.up.pt
Linux moab01 2.6.24-26-server #1 SMP Tue Dec 1 19:19:20 UTC 2009 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
root@moab01:~# mdiag -n
compute node summary
Name                    State   Procs      Memory         Opsys

moab02                   Idle    2:2      1002:1002       linux
-----                     ---    2:2      1002:1002       -----

Total Nodes: 1  (Active: 0  Idle: 1  Down: 0)

root@moab01:~#
```

**Figura 4.3** – Lista dos nós no sistema de teste obtida com o comando 'mdiag –n' do Moab

```
root@moab01:~# showq

active jobs------------------------
JOBID              USERNAME      STATE PROCS   REMAINING            STARTTIME


0 active jobs               0 of 2 processors in use by local jobs (0.00%)
                            0 of 1 nodes active      (0.00%)

eligible jobs----------------------
JOBID              USERNAME      STATE PROCS     WCLIMIT            QUEUETIME


0 eligible jobs

blocked jobs-----------------------
JOBID              USERNAME      STATE PROCS     WCLIMIT            QUEUETIME


0 blocked jobs

Total jobs:  0

root@moab01:~#
```

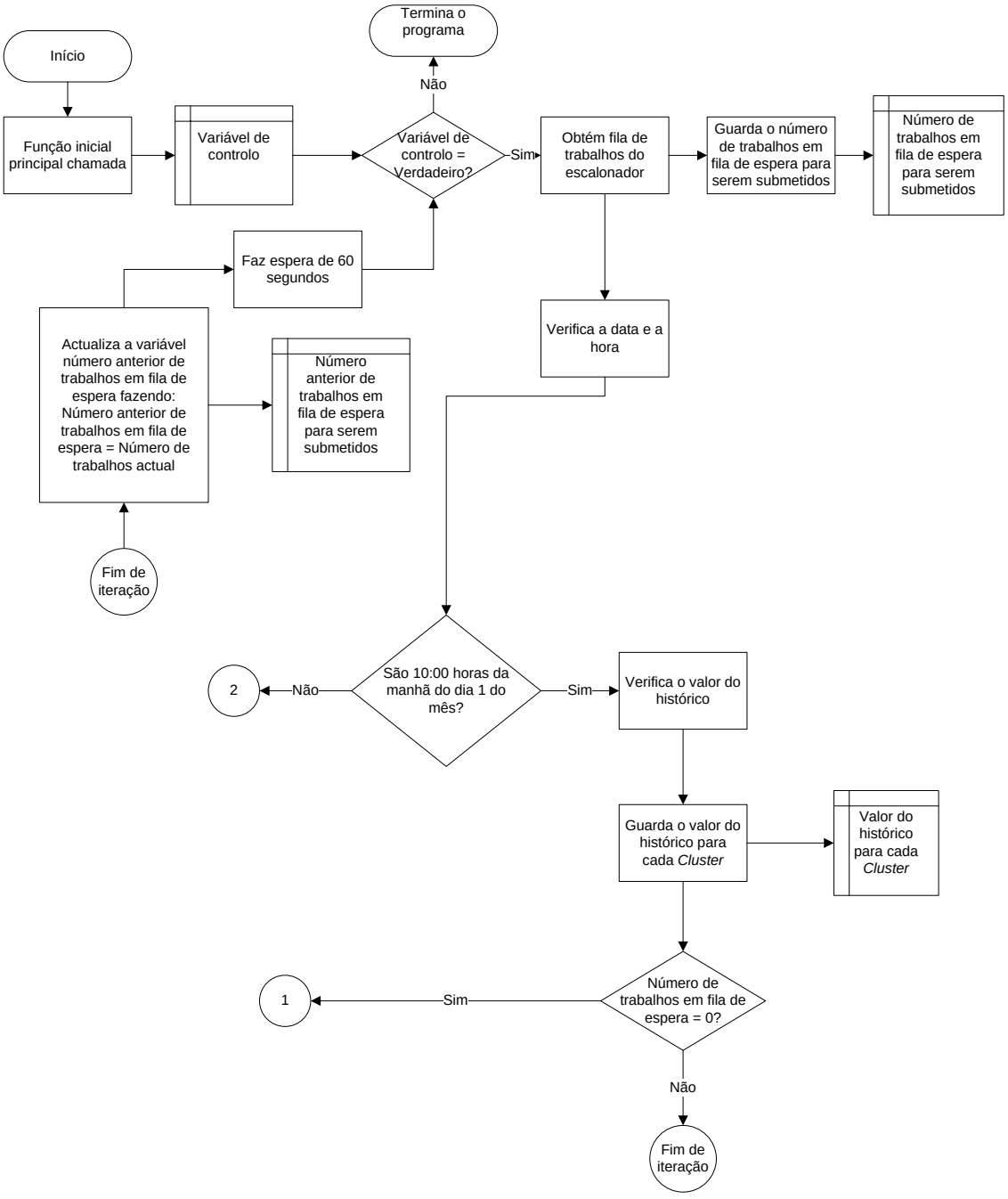
**Figura 4.4** – Fila de trabalhos no sistema de teste obtida com o comando 'showq' do Moab

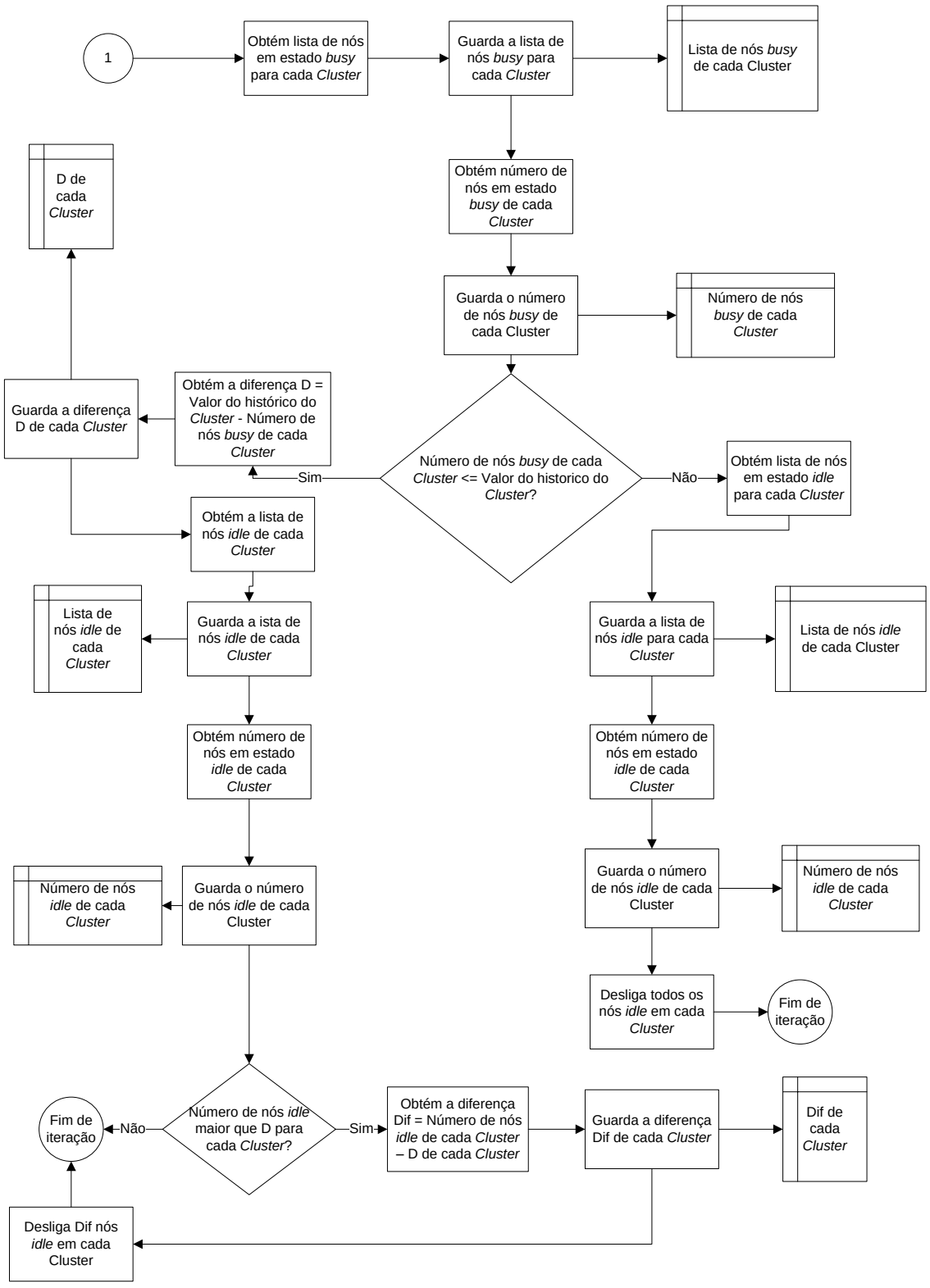
Outra limitação é que nunca chegava a haver um nó para desligar, pois o *Master* tem de estar sempre ligado para gerir os *Slaves* e neste caso, só existindo um *Slave*, mesmo estando este *idle* e não havendo fila de trabalhos, nenhum nó era indicado para ser desligado, pois metade dos nós *idle* seria 0.5 e portanto o *script* neste caso assume o valor mais baixo, logo 0 nós seriam passíveis de um *shutdown*. Depois, outra dificuldade foi encontrada em relação à metodologia inicial. A versão do *software* de escalonamento Moab usada nos *Clusters* da FEUP, a versão *Grid Suite*, não apresenta as capacidades *Green* da versão do Moab instalada no sistema de teste. Logo, na infra-estrutura GridFEUP não há a possibilidade de obter valores instantâneos de temperatura ou potência dos nós a partir do escalonador, logo a parte de optimização do algoritmo, em que em vez de se desligar nós *idle* aleatoriamente se escolheriam os melhores nós a desligar consoante os valores destes parâmetros energéticos não seria possível de implementar e foi posta de lado neste momento. Outra limitação da versão do escalonador instalada na estrutura GridFEUP e dos sistemas de gestão dos servidores desta estrutura é que aparentemente não tem a opção de pôr as máquinas em hibernação ou *stand-by* como se queria na metodologia inicialmente pensada, de forma a evitar os ciclos *shutdown* e *boot*. Logo, o algoritmo foi realizado tendo estes factores em atenção. Apesar de tudo, foi possível chegar a um protótipo inicial que, embora o sistema de teste não permitisse testar todas as suas capacidades, foi possível ensaiar a execução de comandos do Moab no código e se faziam o que era requerido.

A partir deste protótipo inicial, quando a sua optimização estava concluída e mais nada havia a ganhar continuando a executá-lo no sistema de teste, partiu-se para a execução no sistema real. Após os primeiros testes na infra-estrutura GridFEUP foi apreendido um facto que não estava correctamente entendido. De início, pensava-se que na estrutura GridFEUP cada *Cluster* funcionava independentemente, ou seja, cada *Cluster* tinha o seu *Master* que geria os respectivos *Slaves*. Portanto, a versão do protótipo nesta altura era única e sem distinção de *Clusters* no *script*, já que devido ao entendimento que havia da arquitectura do sistema a ideia era instalar este *script* no *Master* de cada *Cluster* para trabalhar em conjunto com o escalonador desse *Cluster*. No entanto, aquando dos testes na infra-estrutura GridFEUP, verificou-se que a área de acesso à estrutura GridFEUP que foi fornecida para testes tinha acesso a todos os *Clusters* de uma vez, permitindo a gestão ainda mais centralizada de toda a infra-estrutura. Por esse facto é que foi preciso fazer ainda novas alterações no código do protótipo, fazendo assim a divisão por *Cluster* evidenciada anteriormente nesta secção. Assim, há a vantagem de a solução ser ainda mais escalável. Sempre que forem feitas alterações à estrutura física, é só alterar um pouco o algoritmo desenvolvido e corrê-lo novamente, podendo até afirmar-se que é por isso bastante rápido e eficaz a sua execução para diferentes sistemas de *Clusters*.

### 4.3.2 – Fluxograma

De seguida apresenta-se um fluxograma relativo ao algoritmo implementado para ilustrar o seu funcionamento. A partir da análise do fluxograma correspondente será mais simples compreender o funcionamento do protótipo implementado, que será descrito na secção seguinte deste capítulo.


Termina o programa
Início
Não
Função inicial principal chamada
Variável de controlo
Variável de controlo = Verdadeiro?
Sim
Obtém fila de trabalhos do escalonador
Guarda o número de trabalhos em fila de espera para serem submetidos
Número de trabalhos em fila de espera para serem submetidos
Faz espera de 60 segundos
Verifica a data e a hora
Actualiza a variável número anterior de trabalhos em fila de espera fazendo: Número anterior de trabalhos em fila de espera = Número de trabalhos actual
Número anterior de trabalhos em fila de espera para serem submetidos
Fim de iteração
São 10:00 horas da manhã do dia 1 do mês?
2
Não
Sim
Verifica o valor do histórico
Guarda o valor do histórico para cada *Cluster*
Valor do histórico para cada *Cluster*
Número de trabalhos em fila de espera = 0?
1
Sim
Não
Fim de iteração


1
Obtém lista de nós em estado *busy* para cada *Cluster*
Guarda a lista de nós *busy* para cada *Cluster*
Lista de nós *busy* de cada Cluster
Obtém número de nós em estado *busy* de cada *Cluster*
Guarda o número de nós *busy* de cada Cluster
Número de nós *busy* de cada *Cluster*
Número de nós *busy* de cada *Cluster* <= Valor do historico do *Cluster*?
Sim
Não
Obtém a diferença D = Valor do histórico do *Cluster* - Número de nós *busy* de cada *Cluster*
Guarda a diferença D de cada *Cluster*
D de cada *Cluster*
Obtém a lista de nós *idle* de cada *Cluster*
Guarda a ista de nós *idle* de cada *Cluster*
Lista de nós *idle* de cada *Cluster*
Obtém número de nós em estado *idle* de cada *Cluster*
Guarda o número de nós *idle* de cada Cluster
Número de nós *idle* de cada *Cluster*
Obtém lista de nós em estado *idle* para cada *Cluster*
Guarda a lista de nós *idle* para cada *Cluster*
Lista de nós *idle* de cada Cluster
Obtém número de nós em estado *idle* de cada *Cluster*
Guarda o número de nós *idle* de cada Cluster
Número de nós *idle* de cada *Cluster*
Desliga todos os nós *idle* em cada *Cluster*
Fim de iteração
Número de nós *idle* maior que D para cada *Cluster*?
Não
Sim
Fim de iteração
Obtém a diferença Dif = Número de nós *idle* de cada *Cluster* – D de cada *Cluster*
Guarda a diferença Dif de cada *Cluster*
Dif de cada *Cluster*
Desliga Dif nós *idle* em cada Cluster

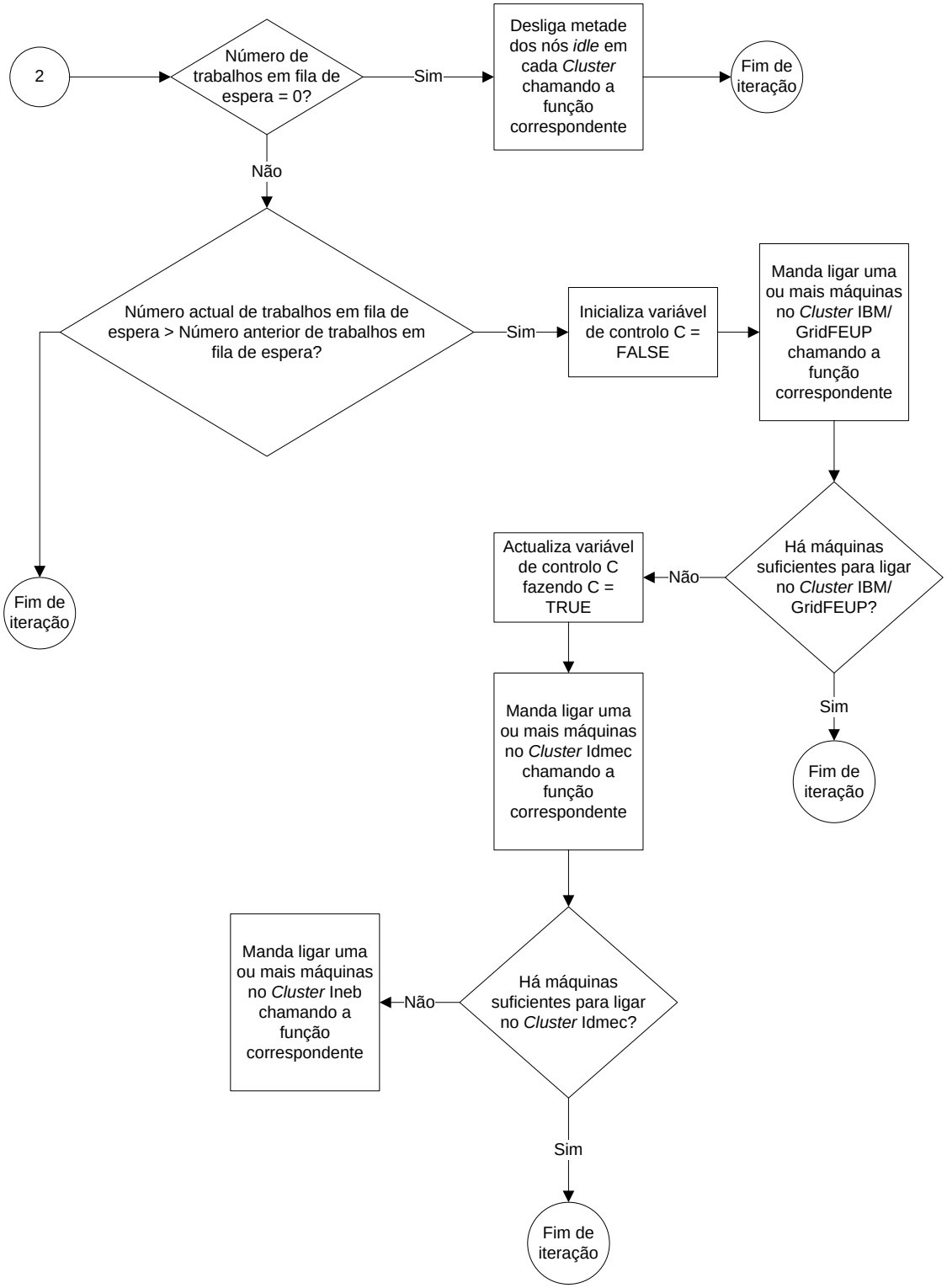



**Figura 4.5** – Fluxograma do algoritmo

A figura 4.6 que se apresenta de seguida mostra a legenda do fluxograma, fazendo luz do significado dos diferentes símbolos do mesmo.

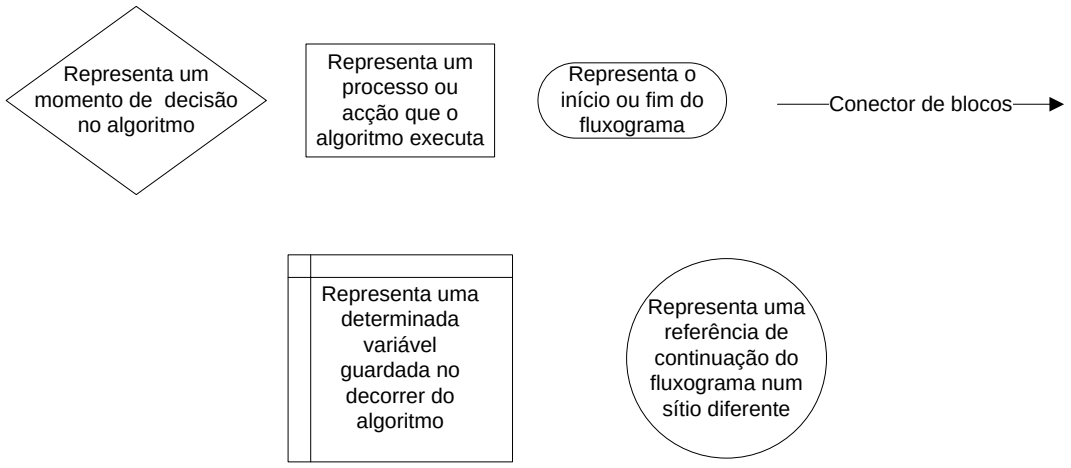



**Figura 4.6** – Legenda do fluxograma

### 4.3.3 – O Algoritmo Implementado

Apresenta-se seguidamente uma explicação sucinta do algoritmo implementado no protótipo.

Como já foi visto, no primeiro dia de cada mês é obtido o valor do histórico para cada *Cluster*. O programa corre num ciclo infinito, executando de 60 a 60 segundos. As linhas seguintes ilustram esta situação:

```
my $valid = TRUE;
while($valid) {
.
.
.
  sleep(60); #Faz espera de 60 segundos
}
```

Dentro deste ciclo 'while' temos todas as chamadas das funções necessárias ao funcionamento do algoritmo. Primeiro, o número de trabalhos em fila de espera para serem executados é obtido a partir de informações do escalonador. Isto é feito com recurso à seguinte função:

```
#OBTÉM NÚMERO DE TRABALHOS NA QUEUE
sub getJobsList() {
  my @date = localtime(time);
```

```
    my $Data = "Teste:".$date[3]." de ".mes($date[4]).", ".$date[2].":".$date[1].":".
$date [0];

    my $outputFile = "jobsList.out";

    my $nj = 0;

    system("mshow -q > $outputFile");#Obtém a fila de trabalhos e direcciona-a
                                                para o ficheiro jobsList.out

    my $FILE = new FileHandle;
    $FILE -> open("<$outputFile") or die "$outputFile is Missing!";

    foreach my $line (<$FILE>) {
        chomp ($line);
        if ($line =~ m/(.*)eligible jobs/) {
            my @aux = split(" ",$line);
            $nj = $aux[0];
        }
    }
    return $nj;
}
```

A função 'getJobsList()' obtém assim a fila de trabalhos para a infra-estrutura GridFEUP com recurso ao comando do Moab 'mshow -q'. Obtido o número de trabalhos em fila de espera, se este número for zero, significando que não há qualquer trabalho à espera de submissão, e se for o primeiro dia do mês, o programa deve então fazer com que sejam mantidos ligados apenas o mesmo número de nós que o histórico indica. Para isso, primeiro é verificado quantos nós estão a fazer "trabalho útil", ou seja, quantos nós estão em estado *busy*. Para verificar quais são estes nós é usada a função 'getNodes()' que recebe um parâmetro do tipo *string* que indica qual o tipo de nós que queremos e retorna o nome dos nós que estão no estado pretendido. Ou seja, se quisermos os nós que estão em estado *busy* a função é chamada como '&getNodes("Busy")', funcionando de forma análoga se quisermos encontrar os nós em estado *Idle* ou *Down*. A função executa outro comando do Moab, o 'mdiag -n', que permite analisar vários aspectos relativos a cada nó de cada *Cluster*, como será visto no capítulo seguinte. Esta função tem a forma seguinte:

```
sub getNodes() {
    my ($nodeState) = @_;

    my @date = localtime(time);

    my $outputFile = "ListaNos.out";
    my @NodesGrid;

    system("mdiag -n > $outputFile");

    my $FH = new FileHandle;
    $FH -> open("<$outputFile") or die "$outputFile is Missing!";

    foreach my $line (<$FH>) {
        chomp $line; #remove \n
        if ($line =~ m/(.*)$nodeState/ and $line =~ m/^node(.*)/ and $line !~ m/^Total(.*)/ and
$line !~ m/(.*)DEFAULT/) {
            my @tempGrid = split(" ",$line);
            push (@NodesGrid, $tempGrid[0]);
        }
    }
    close($FH);
    return @NodesGrid;
}
```

Retomando o algoritmo, encontrados os nós que estão em estado *busy* é guardado este número. Isto servirá para comparar o número de nós que estão a trabalhar com o número de nós que o histórico indica. Aqui há uma decisão. Se o número de nós *busy* for menor ou igual ao valor do histórico, será obtido o valor da diferença D:

D = Valor do histórico - Número de Nós *Busy* (4.1).

Imaginando assim que o valor do histórico era 10 nós e havia 7 nós *busy*, então pela equação (4.1) D seria igual a 3. Depois, obtém-se quais os nós que estão em estado *idle* e guarda-se quantos são. Se o número de nós *idle* for maior que a diferença D, então encontra-se o número de nós N que fazem com que se exceda o valor do histórico. Viria pela equação seguinte:

N = Número de nós *idle* - D (4.2).

Assumindo que havia 5 nós em estado *idle*, N seria igual a 2, e portanto seriam desligados 2

nós *idle*. Se o número de nós *busy* for maior que o valor do histórico, então serão encontrados quais os nós que estão em *idle* e N será igual a esse valor, sendo desligados estes nós. Desta forma apenas permanecem ligados os nós que estão a fazer "trabalho útil" e que mesmo assim já excediam o valor do histórico. Se o número de trabalhos na fila de espera for maior que zero então nada será feito relativamente ao valor do histórico, ficando o *Cluster* na mesma. Isto acontece porque se há trabalhos em fila de espera não faz sentido desligar nós *idle* que depois poderão ser necessários ligar novamente instantes depois para permitir a submissão de trabalhos para aliviar a fila de espera. Para desligar os nós, é usada neste caso a função 'shutdownNodes()'. Esta recebe como parâmetro o valor N obtido e desliga por isso N nós *idle*. Um exemplo da função tem a forma seguinte:

```
sub shutdownNodes() {

    my ($nodesNumber) = @_;

    #1 - Obtém a lista dos nós em Idle
    my @idleNodes = &getNodes("Idle");

    # - Conta quantos estão Idle
    #my $idleNodesSize = @idleNodes;

    #2 - Desliga nós caso estejam em Idle
    if ($nodesNumber > 0) {
        for my $s (1..$nodesNumber) {

            my $idleName = $idleNodes[$s-1];

            system("rpower -n $idleName off");
            #system("mnodectl $idleName -m power=off");
            #system("mnodectl $idleName -m state=Down");
        }
        close($FH);
    }
}
```

De referir que o comando para desligar os nós aqui mostrado - 'rpower -n "nome do nó" off' - é para os servidores IBM, ou seja, neste caso para o *Cluster* IBM/GridFEUP. Para os outros *Clusters* o comando será 'ipmitool -U OEM -P "password" -H "nome do nó" power off'.

O processo descrito acontece sempre às 10:00 horas da manhã do primeiro dia de cada mês. Também de relembrar que apesar de se executar um único *script* na mesma área de

gestão para todos os *Clusters*, este tem no seu código a divisão por *Cluster*, o que faz com que o processo descrito seja executado para cada sistema independentemente.

Após este processo e como o algoritmo executa num ciclo infinito de 60 a 60 segundos, a iteração seguinte será às 10:01 horas da manhã. Desta vez, já não será consultado o histórico e o processo anterior não será feito até às 10:00 horas da manhã do primeiro dia do próximo mês. Assim, será novamente verificada a fila de trabalhos e o novo valor relativo ao número de trabalhos em fila de espera será guardado novamente. Neste ponto haverá novas decisões. Se a fila de trabalhos em espera for igual a zero, serão desligados metade dos nós que estão em estado *idle* em cada *Cluster*. Para isso usa-se a função 'shutdownHalfNodes', cujo essencial se apresenta de seguida:

```
sub shutdownHalfNodes() {
  use integer;

        #1 - Obtém a lista dos nós em Idle
        my @idleNodes = &getNodes("Idle");

        #2 - Conta quantos estão Idle
        my $idleNodesSize = @idleNodes;

        system("mdiag -n >> nodesList.out\n");

        #3 - Desliga metade dos nós caso existam nós em Idle
        if ($idleNodesSize > 0) {
            for my $i (1..$idleNodesSizeGrid/2) {
                my $idleName = $idleNodes[$i-1];
                system("rpower -n $idleName off");
                #system("mnodectl $idleName -m power=off");
                #system("mnodectl $idleName -m state=Down");
            }
        }
}
```

Como se pode verificar, são encontrados quais os nós em estado *idle* e contados quantos são. Depois, metade desses nós são desligados recorrendo aos comandos anteriormente mencionados. Se a fila de trabalhos não for igual a zero há dois casos que podem acontecer. O primeiro é se o número de trabalhos em fila de espera para serem submetidos for superior ao que se verificava na iteração anterior do algoritmo. Neste caso, é possível inferir que as máquinas ligadas nesse instante não sejam suficientes para aliviar a fila de trabalhos em espera para serem submetidos. Portanto, será ligada uma máquina num dos *Clusters*. De notar que neste caso não é ligada uma máquina em cada *Cluster* mas sim uma máquina num

*Cluster* de cada vez. Isto porque pode ser suficiente ligar apenas uma máquina na estrutura GridFEUP para conseguir reduzir o número de trabalhos em espera na fila e como o objectivo é a redução do gasto energético nesta estrutura, não faria sentido ligar de uma vez só mais do que uma máquina sem saber se esta seria efectivamente necessária. O essencial da função usada para ligar uma máquina - 'startMachine' - é mostrado de seguida:

```
sub startMachine() {
  my $numeroProc = &getJobsListP();

  my $numeroNos = ($numeroProc/2) + 1;

  my $downNameGrid = " ";

  my $Control = FALSE;

  my @idleNodes = &getNodes("Idle");
  my $idleNodesSize = @idleNodes;

  my $NN = 0;

  if ($idleNodesSize == 0) {  #Se não existirem máquinas Idle

      #1 - Obtém a lista de nós em Down
      my @downNodes = &getNodes("Down");
      my $downNodesSize = @downNodes;

      #2 - Obtém o nome dos nós em Down necessários
      if ($downNodesSize > 0 and $downNodesSize >= $numeroNos) {
          for my $i (1..$numeroNos) {
              $downName = $downNodes[$i-1];
              3 - Liga esse nó
              system("rpower -n $downName on");
              #system("mnodectl $downName -m power=on");
              #system("mnodectl $downName -m state=Up");
              $Control = TRUE;
          }
      }
      else {
          print ("Não há nós Down suficientes para ligar no Cluster\n");
      }
```

```
    }
    elsif ($idleNodesSize < $numeroNos) {

        $NN = $numeroNos - $idleNodesSize;

        #1 - Obtém a lista de nós em Down
        my @downNodes = &getNodes("Down");
        my $downNodesSize = @downNodes;

        #2 - Obtém o nome dos nós em Down necessários
        if ($downNodesSize > 0 and $downNodesSize >= $NN) {
            for my $i (1..$NN) {
                $downName = $downNodes[$i-1];
                #3 - Liga esse nó
                system("rpower -n $downName on");
                #system("mnodectl $downName -m power=on");
                #system("mnodectl $downName -m state=Up");
                $Control = TRUE;
            }
        }
        else {
            print ("Não há nós Down suficientes para ligar no Cluster\n");
        }
    }
    else {
        print ("Há nós Idle suficientes para correr o trabalho no Cluster\n");
        $Control = TRUE;
    }

    return $Control;
}
```

Como se pode ver pelo código da função apresentado, primeiramente verifica-se qual o número de processadores que o primeiro trabalho da fila de espera requer. Após essa verificação, obtém-se o número de nós em estado *idle* que o *Cluster* tem. Consoante esse número será feita a decisão de ligar ou não nós nesse *Cluster*. Esta forma de agir tem por base a teoria de que se um trabalho necessita por exemplo de quatro processadores para ser executado, se for no *Cluster* IBM/GridFEUP em que cada nó tem dois processadores, seriam precisos ligar dois nós. Ora, se existirem dois ou mais nós *idle* ligados não será necessário

ligar dois servidores que estivessem desligados, assim como se existisse apenas um nó *idle* bastaria ligar uma máquina.

Portanto, se não houverem nós *idle* ligados verifica-se quais e quantos nós *down* existe no *Cluster*. Se o valor destes for maior que o número de nós N necessários para executar o trabalho, sendo N calculado com base nos processadores que o trabalho requer, então serão ligados N nós que estavam desligados. Por exemplo, imaginando que um trabalho necessitava de 6 processadores, no *Cluster* IBM/GridFEUP seria necessário ligar 3 nós se não houvessem nós *idle* no *Cluster*. Se porventura não existissem 3 nós desligados no *Cluster*, seria lançado um sinal de controlo que indica nós insuficientes no primeiro *Cluster* e passar-se-ia ao segundo e assim sucessivamente. É de notar que para o caso de ligar máquinas, mais uma vez a pensar na maior eficiência energética que se quer alcançar, só se ligam nós num *Cluster* em cada iteração. Ou seja, verifica-se se há nós suficientes no primeiro *Cluster* para executar o trabalho que está em espera. Se houver ligam-se esses nós e a iteração acaba aí, se não existirem nós suficientes passa-se ao *Cluster* seguinte onde o mesmo processo ocorre. Assim, difere um pouco do processo de desligar máquinas que em cada iteração ocorre sempre nos três *Clusters*.

Se houver nós *idle* ligados, nesse caso verifica-se se estes são em número suficiente para receber o próximo trabalho da fila de espera. Se não forem, serão então ligados apenas os nós *down* que faltam para atingir o número de nós teoricamente suficiente para executar o trabalho. Se não existirem nós *down* suficientes, mais uma vez é mandado o sinal de controlo desse facto e o mesmo processo ocorre para o *Cluster* seguinte na busca de recursos suficientes nesse sistema para receber o próximo trabalho da fila de trabalhos.

Se houver nós *idle* no *Cluster* e estes forem em número teoricamente suficiente para que haja recursos disponíveis para executar satisfatoriamente o trabalho seguinte da fila, então não se liga nenhum nó.

Os comandos para ligar um nó são semelhantes aos de desligar, sendo para os servidores IBM do *Cluster* IBM/GridFEUP o comando 'rpower -n "nome do nó" on' e para os restantes *Clusters* que usam IPMITools o comando 'ipmitool -U OEM -P "password" -H "nome do nó" power on'.

Se a fila de trabalhos em espera para submissão for maior do que zero mas o número de trabalhos em fila de espera for menor ou igual do que na iteração anterior do *script* então não mudará nada no *Cluster*, pois é justo inferir que nesse caso os nós que estão ligados estão a conseguir aliviar a carga de trabalhos em espera no escalonador.

Este processo será assim repetido infinitamente de 60 em 60 segundos. No capítulo seguinte mostra-se os testes efectuados e a partir dos resultados obtidos faz-se um estudo da viabilidade ou não da solução implementada.

# Capítulo 5

# Resultados e Discussão

## 5.1 – Introdução

Este capítulo começa por descrever os testes efectuados no protótipo implementado com base no modelo idealizado e explicitado no capítulo anterior. Posteriormente apresentam-se os resultados obtidos e far-se-á uma breve discussão dos mesmos culminando o capítulo com uma análise critica à viabilidade do modelo de solução encontrado para o problema.

## 5.2 – Testes

Para obter alguns resultados práticos passíveis de análise posterior testou-se o protótipo na infra-estrutura GridFEUP como inicialmente previsto. Para isso, foi garantido o acesso à infra-estrutura através de um cliente *ssh* simples, alocando-se uma área de trabalho que permitiu estudar várias funcionalidades da versão do Moab instalada na GridFEUP. Posteriormente estudaram-se os comandos adequados a esta versão para o correcto funcionamento do protótipo. Depois de feito o *login*, acede-se à zona local da GridFEUP que foi disponibilizada para os testes, sendo que a imagem seguinte representa a interface para o utilizador dessa zona.

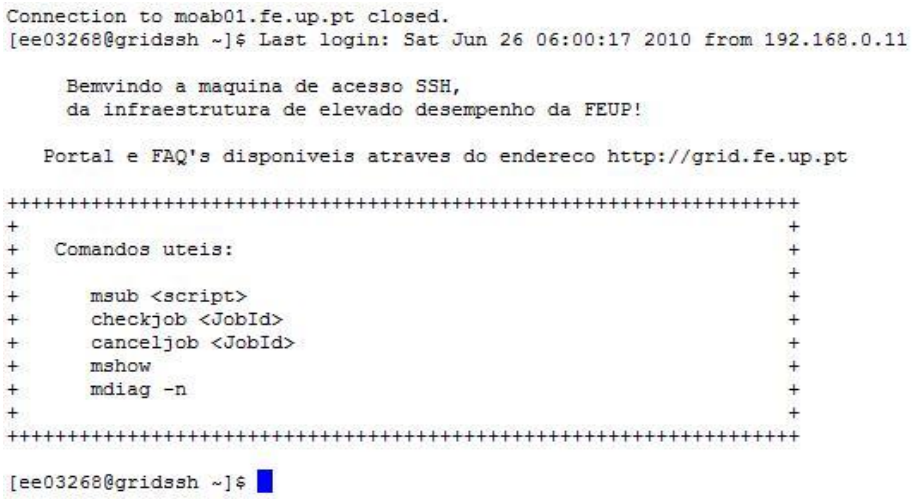



**Figura 5.1** – Interface para o utilizador da área de trabalho alocada para testes na infra-estrutura GridFEUP

Os vários comandos do Moab ensaiados permitem ver o estado dos nós, da fila de espera, submeter trabalhos, cancelar trabalhos em execução, desligar máquinas, etc, naturalmente desde que existam as permissões apropriadas para o fazer. Isto permitiu testar algumas capacidades do Moab e a partir dos resultados observados refinar o código do *script* efectuado. Foi nesta altura, por exemplo, que se verificou que toda a estrutura GridFEUP pode ser gerida a partir deste nó, o que originou as mudanças no *script* referidas no capítulo anterior.

Como foi visto no capítulo anterior, o resultado de dois comandos do Moab já mencionados previamente neste trabalho e bastante importantes para o desenvolvimento do *script* efectuado, como o são o ‘mdiag –n’ e o ‘mshow -q’, permite obter informações importantes relativamente aos nós do *Cluster*. Tem-se então:

- O comando 'mdiag -n', que fornece informação detalhada sobre o estado dos nós que o Moab está a acompanhar no momento. É possível ver que este comando mostra uma lista de todos os nós dos três diferentes *Clusters* da FEUP. No apêndice C, os resultados aí detalhados mostram a lista de nós que se obtém com o comando ‘mdiag –n’ na infra-estrutura GridFEUP. Os primeiros 31 nós que aparecem na lista, com o nome "nodexy" com xy = [1,3...32] são os nós pertencentes ao *Cluster* IBM/GridFEUP. Os seguintes 39 nós são os que pertencem ao *Cluster* Idmec, tendo estes o nome "idmecaxy" com xy = [1...20] e "idmecbxy" com xy = [1...19]. Os restantes 18 nós pertencem ao *Cluster* Ineb e apresentam o nome "inebxy" com xy = [1...18]. De referir que o *Cluster* Idmec apresenta nós com o nome "idmeca" e "idmecb" já que este *Cluster* tem 2 bastidores e desta forma permite-se mais facilmente localizar o nó em causa. O comando 'mdiag -n' permite ainda: Aquilatar do estado de cada nó, se está *busy,idle, down, running*, etc; verificar, na coluna *"Procs"*, quantos processadores tem cada nó e quantos estão no momento ocupados a executar algum trabalho; verificar qual o sistema operativo a correr nos nós; ver qual a memória em MB de cada nó. É a partir deste *output* que o *script* vai buscar quais os nós que estão em determinado estado e qual o seu nome de forma a poder ligá-los ou desligá-los. É ainda importante notar que houve preocupação em excluir o nó *"DEFAULT"* dos processos que são accionados no *script*. Como

se pode ver pelo resultado do comando existe um nó deste género em cada *Cluster*, que está sempre desligado e que serve para as actividades de gestão do *Cluster*, logo não deve entrar nas "contas" do *script*.

- O comando 'mshow -q' permite ver a lista de todos os trabalhos que o escalonador tem sob a sua gestão. Como se pode observar no seu resultado exposto no Apêndice C, este comando lista todos os trabalhos que o escalonador está a gerir no momento, aparecendo assim na coluna *"State"* como *running*. Estando em fila de espera, os trabalhos surgem com o estado de *eligible* nessa coluna e se surgirem como bloqueados, ou seja, que pararam de executar, na coluna aparece referida a razão de terem sido parados sem terminar. Outras informações adicionais podem ser encontradas no *output* deste comando, como por exemplo: O ID, o utilizador que o submeteu, o número de processadores que estão a ser usados para executar o trabalho ou que o trabalho, em princípio, necessita para poder ser executado; a data de início e uma estimativa de quanto demorará. É a partir deste *output* que o *script* obtém o número de trabalhos que estão em fila de espera para serem executados e o número de processadores que cada trabalho requer. Estas informações são apenas referentes aos trabalhos em estado *eligible*, que são os que estão em fila de espera.

Neste ponto tem-se toda a informação necessária para executar o algoritmo desenvolvido. Para isso basta estar com o *login* feito na zona ilustrada anteriormente e chamar o algoritmo como mostrado na figura seguinte:

```
[ee03268@gridssh ~]$ ./algoritmo5.pl
main::verify() called too early to check prototype at ./algoritmo5.pl line 7.
```

**Figura 5.2** – Interface de teste com a chamada do algoritmo na linha de comandos

O algoritmo fica assim a correr infinitamente e não perturba o normal funcionamento do sistema.

Um factor importante a referir em relação aos testes é que se optou por correr o programa e enviar os resultados para dois ficheiros (o nodesList.out e o Queues.out) cujos resultados serão apresentados na próxima secção deste capítulo. O primeiro recebe em cada iteração do algoritmo o resultado do 'mdiag -n' desse instante e calcula depois, consoante o caso, quais os nós a ligar ou desligar em cada *Cluster* ou se não faz nada, gravando também a data e hora de cada teste. O segundo ficheiro grava também a data e hora do teste e em cada iteração acrescenta ao ficheiro qual a fila de trabalhos do *Cluster* correspondente nesse instante.

Como nota prévia antes da apresentação dos resultados, o formato de apresentação dos mesmos nos ficheiros nodesList.out e Queues.out é na realidade o apresentado em detalhe no exemplo **1** do Apêndice C. Nesta secção, para permitir uma mais rápida e fácil compreensão dos resultados, estes são estruturados de forma algo diferente.

## 5.3 – Resultados e Discussão

Nesta secção serão apresentados e discutidos alguns dos resultados obtidos durante os dois dias de teste nas infra-estruturas GridFEUP. De seguida serão expostos os resultados de três iterações diferentes, portanto de três alturas diferentes nesse período.

**1**

**Primeiro Momento**

**nodesList.out**

Teste:10 de Junho, 10:3:27

compute node summary

*Cluster* IBM/GridFEUP:

Nós *Busy* - Total = 2: node01; node03

Nós *Idle* - Total = 28: node32; node31; node30; node29; node28; node27; node26; node25; node24; node22; node21; node20; node19; node18; node17; node16; node15; node14; node13; node12; node11; node10; node09; node08; node07; node 06; node 05; node 04

Nós *Down* - Total = 1: node23

*Cluster* Idmec:

Nós *Busy* - Total = 24: idmecb19; idmecb18; idmecb17; idmecb16; idmecb15; idmecb14; idmecb13; idmecb12; idmecb11; idmecb09; idmecb08; idmecb06; idmecb03; idmecb02; idmecb01; idmeca09; idmeca08; idmeca07; idmeca06; idmeca05; idmeca04; idmeca03; idmeca02; idmeca01

Nós *Idle* - Total = 12: idmecb05; idmecb04; idmeca20; idmeca18; idmeca17; idmeca16; idmeca15; idmeca14; idmeca13; idmeca12; idmeca11; idmeca10

Nós *Down* - Total = 3: idmecb10; idmecb07; idmeca19

*Cluster* Ineb:

Nós *Busy* - Total = 0

Nós *Idle* - Total = 17: ineb18; ineb17; ineb16; ineb15; ineb14; ineb13; ineb12; ineb11; ineb10; ineb09; ineb08; ineb07; ineb06; ineb05; ineb04; ineb03; ineb02

Nós *Down* - Total = 1: ineb01

Total Nodes: 88 (Active: 26 Idle: 57 Down: 5)

Teste:10 de Junho, 10:3:27
Os nos a desligar do Cluster GridFEUP/IBM serao:
node32
node31
node30
node29
node28
node27
node26
node25
node24
node22
node21
node20
node19
node18
Teste:10 de Junho, 10:3:27
Os nos a desligar do Cluster Idmec serao:
idmecb05
idmecb04
idmeca20
idmeca18
idmeca17
idmeca16
Teste:10 de Junho, 10:3:27
Os nos a desligar do Cluster Ineb serao:
ineb18
ineb17
ineb16
ineb15
ineb14
ineb13
ineb12
ineb11

**Queues.out**
Teste:10 de 5, 10:3:27

eligible jobs----------
JOBID USERNAME STATE PROCS WCLIMIT QUEUETIME

0 eligible jobs

**Segundo Momento**

**nodesList.out**

Teste:10 de Junho, 13:4:20

compute node summary

*Cluster* IBM/GridFEUP:

Nós *Busy* - Total = 2: node01; node03

Nós *Idle* - Total = 28: node32; node31; node30; node29; node28; node27; node26; node25; node24; node22; node21; node20; node19; node18; node17; node16; node15; node14; node13; node12; node11; node10; node09; node08; node07; node 06; node 05; node 04

Nós *Down* - Total = 1: node23

*Cluster* Idmec:

Nós *Busy* - Total = 30: idmecb19; idmecb18; idmecb17; idmecb16; idmecb15; idmecb14; idmecb13; idmecb12; idmecb11; idmecb09; idmecb08; idmecb06; idmecb03; idmecb02; idmecb01; idmeca15; idmeca14; idmeca13; idmeca12; idmeca11; idmeca10; idmeca09; idmeca08; idmeca07; idmeca06; idmeca05; idmeca04; idmeca03; idmeca02; idmeca01

Nós *Idle* - Total = 6: idmecb05; idmecb04; idmeca20; idmeca18; idmeca17; idmeca16

Nós *Down* - Total = 3: idmecb10; idmecb07; idmeca19

*Cluster* Ineb:

Nós *Busy* - Total = 0

Nós *Idle* - Total = 17: ineb18; ineb17; ineb16; ineb15; ineb14; ineb13; ineb12; ineb11; ineb10; ineb09; ineb08; ineb07; ineb06; ineb05; ineb04; ineb03; ineb02

Nós *Down* - Total = 1: ineb01

Total Nodes: 88  (Active: 32  Idle: 51  Down: 5)

Teste:10 de Junho, 13:4:20

Os nos a desligar do Cluster GridFEUP/IBM serao:

node32

node31

node30

node29

node28

node27

node26

node25
node24
node22
node21
node20
node19
node18
Teste:10 de Junho, 13:4:20
Os nos a desligar do Cluster Idmec serao:
idmecb05
idmecb04
idmeca20
Teste:10 de Junho, 13:4:20
Os nos a desligar do Cluster Ineb serao:
ineb18
ineb17
ineb16
ineb15
ineb14
ineb13
ineb12
ineb11

**Queues.out**
Teste:10 de 5, 13:4:20

eligible jobs----------
JOBID USERNAME STATE PROCS WCLIMIT QUEUETIME

0 eligible jobs

**Terceiro Momento**
**nodesList.out**
Teste:10 de Junho, 13:15:27
compute node summary

*Cluster* IBM/GridFEUP:

Nós *Busy* - Total = 2: node01; node03

Nós *Idle* - Total = 28: node32; node31; node30; node29; node28; node27; node26; node25; node24; node22; node21; node20; node19; node18; node17; node16; node15; node14; node13; node12; node11; node10; node09; node08; node07; node 06; node 05; node 04

Nós *Down* - Total = 1: node23

*Cluster* Idmec:

Nós *Busy* - Total = 34: idmecb19; idmecb18; idmecb17; idmecb16; idmecb15; idmecb14; idmecb13; idmecb12; idmecb11; idmecb09; idmecb08; idmecb06; idmecb03; idmecb02; idmecb01; idmeca20; idmeca18;idmeca17; idmeca16; idmeca15; idmeca14; idmeca13; idmeca12; idmeca11; idmeca10; idmeca09; idmeca08; idmeca07; idmeca06; idmeca05; idmeca04; idmeca03; idmeca02; idmeca01

Nós *Idle* - Total = 2: idmecb05; idmecb04

Nós *Down* - Total = 3: idmecb10; idmecb07; idmeca19

*Cluster* Ineb:

Nós *Busy* - Total = 0

Nós *Idle* - Total = 17: ineb18; ineb17; ineb16; ineb15; ineb14; ineb13; ineb12; ineb11; ineb10; ineb09; ineb08; ineb07; ineb06; ineb05; ineb04; ineb03; ineb02

Nós *Down* - Total = 1: ineb01

Total Nodes: 88 (Active: 36 Idle: 47 Down: 5)

Teste:10 de Junho, 13:15:27
Os nos a desligar do Cluster GridFEUP/IBM serao:
node32
node31
node30
node29
node28
node27
node26
node25
node24
node22
node21
node20
node19
node18
Teste:10 de Junho, 13:15:27
Os nos a desligar do Cluster Idmec serao:

```
idmecb05
Teste:10 de Junho, 13:15:27
Os nos a desligar do Cluster Ineb serao:
ineb18
ineb17
ineb16
ineb15
ineb14
ineb13
ineb12
ineb11

Queues.out
Teste:10 de 5, 13:15:27

eligible jobs----------------------
JOBID             USERNAME      STATE PROCS    WCLIMIT         QUEUETIME


0 eligible jobs
```

Destes três exemplos dos resultados obtidos a primeira análise retirada é a contínua repetição do valor zero para o número de trabalhos em fila de espera. Isto faz com que o algoritmo entre sempre na mesma função, a 'shutdownHalfNodes()' que desliga metade dos nós *idle* em cada *Cluster*. Deste modo, pode-se concluir pelos testes efectuados que o algoritmo funciona bem. Veja-se que por exemplo no primeiro momento exibido, há 28 nós *idle* no *Cluster* IBM/GridFEUP e depois é dito pelo programa que há 14 nós a desligar, correspondendo esses 14 indicados a metade dos nós *idle* deste *Cluster*. Para o *Cluster* Idmec observa-se que há 12 nós *idle* e que são indicados 6 desses 12 nós para serem desligados. Já relativamente ao *Cluster* Ineb, observa-se que há 17 nós *idle*, o que faria com que ao desligar metade se desligassem 8.5 nós, o que obviamente não é possível. Nestes casos, o *script* decide pela aproximação inferior, ou seja, neste caso seriam 8 nós deste *Cluster* a desligar, que é o que se verifica.

Nos dois momentos seguintes, à medida que vão aumentando os nós activos e diminuindo os nós *idle*, verifica-se o correcto funcionamento do programa ao reagir a mudanças no estado do *Cluster*. Assim, e como a lista de trabalhos em espera para serem submetidos continua a ser zero, o programa está na mesma a entrar no caso de desligar metade dos nós *idle*. Verifica-se no segundo momento por exemplo, que ao diminuir o número de nós *idle*, o algoritmo implementado tem isso em atenção e também diminui o número de nós que indica para desligar. Adicionalmente, verifica-se que o número de nós *idle* que diminuiu foi no *Cluster* Idmec, de 12 no primeiro momento para 6 neste. Mais uma vez se ilustra o bom funcionamento do algoritmo, que indica agora apenas 3 nós para desligar neste *Cluster*, correctamente escolhidos de entre os 6 possíveis. Um funcionamento análogo é observado no terceiro momento.

Após estes dois dias de testes verificou-se sempre um correcto funcionamento e sem falhas do protótipo implementado. No entanto, a fila de trabalhos em espera para serem submetidos foi sempre zero neste período. Desta forma, optou-se por parar os testes e retomá-los periodicamente nos dias seguintes para observar se havia mudanças no valor de trabalhos em espera na fila. Infelizmente, isso nunca aconteceu e esse valor foi sempre igual a zero, o que fazia com que o protótipo entrasse sempre na mesma função e fizesse sempre a mesma coisa. Assim, foi tomada a decisão de "forçar" o algoritmo a entrar noutra função, como a de ligar máquinas, para atestar da sua funcionalidade exacta ou não. Para esse efeito, criou-se um ficheiro com um formato igual ao da fila de trabalhos originada pelo comando 'mshow -q' do Moab. Esse ficheiro era lido como se fosse gerado pelo programa embora fosse agora manipulado manualmente. Deste modo, inventaram-se trabalhos que estariam em estado *eligible*, ou seja, em fila de espera no escalonador e adicionados ao sítio correcto no ficheiro que representava agora a fila de trabalhos lida em cada iteração. O ficheiro tinha agora o aspecto seguinte, com a linha do trabalho adicionado à fila representada a negrito e itálico:

**Ficheiro de teste da fila de espera manipulado:**

```
active jobs------------------------
JOBID              USERNAME      STATE PROCS   REMAINING          STARTTIME

8094               mashar    Running     8 99:10:25:12  Wed Jun 23 10:31:31
8127               deq05006  Running     4 99:17:55:30  Wed Jun 23 18:01:49

2 active jobs           12 of 272 processors in use by local jobs (4.41%)
                         6 of 83 nodes active      (7.23%)

eligible jobs----------------------
JOBID              USERNAME      STATE PROCS     WCLIMIT          QUEUETIME

ibm.7289                  aba       Idle         4  99:23:59:59Thu Jan 14 15:51:31

1 eligible jobs

blocked jobs-----------------------
JOBID              USERNAME      STATE PROCS     WCLIMIT          QUEUETIME

ibm.7289               aba  BatchHold     4 99:23:59:59  Thu Jan 14 15:51:31
ibm.7296               aba  BatchHold     4 99:23:59:59  Thu Jan 14 16:41:02

2 blocked jobs

Total jobs:  4
```

Desta forma, ao executar o programa já haveria trabalhos em fila de espera e depois era só aumentar ou diminuir o número de trabalhos para verificar a correcção do algoritmo. Entre cada iteração o método era manipular o ficheiro e gravá-lo na zona correspondente para que

na iteração seguinte, 60 segundos depois, o *script* lesse o ficheiro alterado e permitia assim analisar a exactidão da sua execução. Seguidamente apresenta-se os resultados obtidos para uma série de testes seguidos com estas manipulações, sendo que do ficheiro de teste da fila de espera só se mostra a parte referente ao *eligible jobs*, que é a informação lida pelo programa e que é manipulada.

Primeiro, usou-se a seguinte lista de nós (obtida com o comando 'mdiag -n' na realidade):

**Ficheiro de teste da lista de nós:**

compute node summary

*Cluster* IBM/GridFEUP:

Nós *Busy* - Total = 6: node32; node31; node19; node18; node17; node16

Nós *Idle* - Total = 3: node28; node26; node24

Nós *Down* - Total = 22: node30; node29; node27; node25; node23; node22; node21; node20; node15; node14; node13; node12; node11; node10; node09; node08; node07; node06; node05; node04; node03; node01

*Cluster* Idmec:

Nós *Busy* - Total = 19: idmecb09; idmecb08; idmecb06; idmecb05; idmecb04; idmecb03; idmecb02; idmecb01; idmeca20; idmeca18; idmeca17; idmeca16; idmeca15; idmeca14; idmeca12; idmeca10; idmeca09; idmeca05; idmeca01

Nós *Idle* - Total = 13: idmecb18; idmecb17; idmecb16; idmecb14; idmecb13; idmecb11; idmeca13; idmeca11; idmeca07; idmeca06; idmeca04; idmeca03; idmeca02

Nós *Down* - Total = 7: idmecb19; idmecb15; idmecb12; idmecb10; idmecb07; idmeca19; idmeca08

*Cluster* Ineb:

Nós *Busy* - Total = 0

Nós *Idle* - Total = 17: ineb18; ineb17; ineb16; ineb15; ineb14; ineb13; ineb12; ineb11; ineb10; ineb09; ineb08; ineb07; ineb06; ineb05; ineb04; ineb03; ineb02

Nós *Down* - Total = 1: ineb01

```
Total Nodes: 88  (Active: 26  Idle: 33  Down: 30)
```

A primeira iteração tinha então o seguinte conteúdo no ficheiro que simulava a fila de trabalhos do escalonador (obtida com o comando 'mshow -q' na realidade):

**Ficheiro de teste da fila de espera - 1ª iteração:**

```
eligible jobs----------------------
JOBID              USERNAME      STATE PROCS    WCLIMIT         QUEUETIME


0 eligible jobs
```

Portanto, na primeira iteração o *script* deveria retornar quais os nós *idle* a desligar em cada *Cluster*. Este foi o resultado obtido:

```
Teste:24 de Junho, 20:36:11
Os nos a desligar do Cluster GridFEUP/IBM serao:
node28
Teste:24 de Junho, 20:36:11
Teste:24 de Junho, 20:36:11
Os nos a desligar do Cluster Idmec serao:
idmecb18
idmecb17
idmecb16
idmecb14
idmecb13
idmecb11
Teste:24 de Junho, 20:36:11
Os nos a desligar do Cluster Ineb serao:
ineb18
ineb17
ineb16
ineb15
ineb14
ineb13
ineb12
ineb11
```

Como se observa o *script* funcionou perfeitamente. Entretanto, alterou-se o ficheiro da fila de trabalhos, acrescentando-lhe agora um trabalho em fila de espera:

**Ficheiro de teste da fila de espera - 2ª iteração:**

```
eligible jobs----------------------
JOBID              USERNAME      STATE PROCS    WCLIMIT         QUEUETIME
```

```
ibm.7289            aba             Idle        21   99:23:59:59  Thu Jan 14 15:51:31

1 eligible jobs
```

Agora já havia um trabalho em fila de espera na fila de trabalhos. O *script* retornou neste caso:

```
Teste:24 de Junho, 20:37:11
Os nos a ligar do Cluster GridFEUP/IBM serao:
node30
node29
node27
node25
node22
node21
node20
node15
```

Analisando este resultado, verifica-se que o programa funcionou correctamente. Com a adição do novo trabalho à fila, o programa verificou que o valor da fila de espera era maior que na iteração anterior e entrou no ciclo de ligação de nós. Desta forma, e de acordo com o que já foi dito que só serão ligados nós de um *Cluster*, os suficientes para permitir a submissão do trabalho, o *script* foi verificar em que *Cluster* havia mais nós disponíveis. Optou-se pela ordem que os *Clusters* aparecem na lista de nós, ou seja, primeiro o IBM/GridFEUP é verificado, se não tiver nós suficientes é verificado o Idmec, se este não tiver também recursos suficientes disponíveis verifica-se o Ineb. De acordo com a lista de nós usada para estes testes, que foi sempre aquela apresentada aquando do início desta ronda de testes, verifica-se que no *Cluster* IBM/GridFEUP há 22 nós em estado *down*. Observa-se também pela fila de espera que o trabalho necessita de 21 processadores para ser executado, por isso necessita de N nós para poder ser submetido, sendo que N vem:

N = (Número de processadores / Processadores por nó) +1 (5.1),

logo pela equação (5.1) vem: N = (inteiro de (21 / 2)) + 1 <=> N = 10 +1 = 11. Neste caso soma-se mais um nó pois o número de processadores é ímpar. Logo, nesta situação eram necessários 10.5 nós ligados e portanto, como não é possível mandar ligar apenas um processador de determinado servidor, soma-se mais um nó já que o programa assume a aproximação inferior para o resultado de 21 / 2. Se o número fosse par daria o número de nós certos que eram precisos que estivessem ligados e não se somaria mais um nó nesse caso. Esse cuidado foi tido no desenvolvimento do protótipo. Assim, verifica-se que são precisos 11 nós ligados neste *Cluster*. O número NL de nós a ligar será então:

NL = N - número de nós *idle* no *Cluster* (5.2).

Na lista de nós observa-se que há 3 nós *idle* ligados neste sistema, logo pela equação (5.2) viria:

NL = 11 - 3 <=> NL = 8.

Seria necessário ligar 8 nós que estejam *down* no *Cluster* IBM/GridFEUP. Como o resultado indica, o *script* deu certo pois retornou uma lista de 8 nós que se encontram *down* para serem ligados.

Na iteração seguinte acrescentaram-se ainda mais dois trabalhos à fila de espera. O ficheiro de teste era o seguinte:

**Ficheiro de teste da fila de espera - 3ª iteração:**

```
eligible jobs----------------------
JOBID           USERNAME      STATE PROCS    WCLIMIT         QUEUETIME

ibm.7289          aba           Idle       9   99:23:59:59  Thu Jan 14 15:51:31
ibm.7289          aba           Idle      21   99:23:59:59  Thu Jan 14 15:51:31
ibm.7289          aba           Idle       6   99:23:59:59  Thu Jan 14 15:51:31

3 eligible jobs
```

O algoritmo deveria fazer novamente o processo descrito anteriormente. Neste caso, o trabalho em espera precisava de 9 processadores, logo necessitaria de N = (inteiro(9/2)) + 1 = 5 nós ligados para poder ser submetido para execução neste *Cluster*. Portanto, como se usou a mesma lista de nós em toda esta ronda de testes, foi visto que havia 3 nós *idle* no *Cluster*, logo para os 5 nós ligados e disponíveis necessários à execução do trabalho em fila de espera, eram necessários ligar mais dois nós que estavam *down* no *Cluster*. O programa retornou o seguinte:

```
Teste:24 de Junho, 20:38:11
Os nos a ligar do Cluster GridFEUP/IBM serao:
node30
node29
```

Portanto, mais uma vez o teste provou o bom funcionamento do algoritmo.

Nas iterações seguintes não se alterou nada no ficheiro de teste da fila de espera, continuando o mesmo número de trabalhos na fila de espera e o programa retornou agora:

```
Não muda nada
Não muda nada
```

Este *output* foi implementado para provar que realmente nada acontece no programa neste caso, como se observou.

Por fim, voltou a colocar-se a fila sem trabalhos em espera para submissão, tendo o *script*

reagido mais uma vez correctamente a essa mudança de estado da fila de espera, retornando quais os nós que podem ser agora desligados em cada *Cluster*, como se pode ver de seguida:

```
Teste:24 de Junho, 20:41:11
Os nos a desligar do Cluster GridFEUP/IBM serao:
node28
Teste:24 de Junho, 20:41:11
Teste:24 de Junho, 20:41:11
Os nos a desligar do Cluster Idmec serao:
idmecb18
idmecb17
idmecb16
idmecb14
idmecb13
idmecb11
Teste:24 de Junho, 20:41:11
Os nos a desligar do Cluster Ineb serao:
ineb18
ineb17
ineb16
ineb15
ineb14
ineb13
ineb12
ineb11
```

Só faltava agora testar se por exemplo na eventualidade de não haver nós disponíveis (soma de nós *down* com os *idle*) suficientes para ligar no *Cluster* que está a ser verificado para possível aceitação de trabalhos, o algoritmo iria verificar correctamente o próximo *Cluster* e assim sucessivamente. Para esta nova ronda de testes, usou-se a seguinte lista de nós (obtida com o comando 'mdiag -n' na realidade):

**Ficheiro de teste da lista de nós:**

compute node summary

*Cluster* IBM/GridFEUP:

Nós *Busy* - Total = 27: node32; node31; node30; node29; node27; node23; node22; node21; node20; node19; node18; node17; node16; node15; node14; node13; node12; node11; node10; node09; node08; node07; node06; node05; node04; node03; node01

Nós *Idle* - Total = 2: node28; node24

Nós *Down* - Total = 2: node26; node25

*Cluster* Idmec:

Nós *Busy* - Total = 31: idmecb19; idmecb18; idmecb16; idmecb15; idmecb14; idmecb11; idmecb09; idmecb08; idmecb06; idmecb05; idmecb04; idmecb03; idmecb02; idmecb01; idmeca20; idmeca18; idmeca17; idmeca16; idmeca15; idmeca14; idmeca13; idmeca12; idmeca11; idmeca10; idmeca09; idmeca08; idmeca07; idmeca06; idmeca05; idmeca04; idmeca01

Nós *Idle* - Total = 2: idmeca03; idmeca02

Nós *Down* - Total = 6: idmecb17; idmecb13; idmecb12; idmecb10; idmecb07; idmeca19

*Cluster* Ineb:

Nós *Busy* - Total = 0

Nós *Idle* - Total = 17: ineb18; ineb17; ineb16; ineb15; ineb14; ineb13; ineb12; ineb11; ineb10; ineb09; ineb08; ineb07; ineb06; ineb05; ineb04; ineb03; ineb02

Nós *Down* - Total = 1: ineb01

Total Nodes: 88 (Active: 58 Idle: 21 Down: 9)

O ficheiro de teste da fila de espera agora usado foi:

**Ficheiro de teste da fila de espera - 1ª iteração:**

```
eligible jobs----------------------
JOBID              USERNAME      STATE PROCS     WCLIMIT           QUEUETIME


0 eligible jobs
```

Na primeira iteração o programa retornou o que era esperado, ou seja, sem nenhum trabalho em fila de espera, o *output* do *script* foi a lista dos nós *idle* a desligar:

```
Teste:25 de Junho, 0:39:53
Os nos a desligar do Cluster GridFEUP/IBM serao:
node28
Teste:25 de Junho, 0:39:53
Teste:25 de Junho, 0:39:53
Os nos a desligar do Cluster Idmec serao:
idmeca03
Teste:25 de Junho, 0:39:53
```

```
Os nos a desligar do Cluster Ineb serao:
ineb18
ineb17
ineb16
ineb15
ineb14
ineb13
ineb12
ineb11
```

Para a segunda iteração do programa mudou-se o ficheiro de teste da fila de espera, acrescentando-se agora um trabalho em espera para submissão, como se vê de seguida:

**Ficheiro de teste da fila de espera - 2ª iteração:**

```
eligible jobs----------------------
JOBID              USERNAME      STATE PROCS    WCLIMIT         QUEUETIME

ibm.7289           aba           Idle       6   99:23:59:59  Thu Jan 14 15:51:31

1 eligible jobs
```

Como se pode ver o trabalho agora acrescentado em fila de espera requer 6 processadores para ser executado. O resultado depois desta alteração na fila de trabalhos foi o seguinte:

```
Teste:25 de Junho, 0:40:53
Os nos a ligar do Cluster GridFEUP/IBM serao:
node26
```

Como se pode observar, o algoritmo retornou um único nó para ligar. Pela lista de nós que foi usada nesta ronda de testes, verifica-se que o *Cluster* tem dois nós *idle* ligados. Para aquilatar da disponibilidade deste sistema receber o trabalho em espera, o *script* executa o mesmo cálculo mencionado anteriormente: divide o número de processadores que o trabalho requer por 2, pois o *Cluster* IBM/GridFEUP tem dois processadores por nó, o que torna claro que o trabalho requer 3 nós deste *Cluster*; posteriormente verifica quantos nós *idle* há neste sistema, que no caso são 2; faz o cálculo de quantos nós NL necessita ligar para colmatar o défice de recursos ligados no sistema, vindo para esta situação, pela equação (5.2), NL = 3 - 2 = 1, o que quer dizer que tem de haver pelo menos um nó *down* neste *Cluster* para este poder receber e executar o trabalho em espera. Portanto, como há dois nós desligados no *Cluster*, é retornado o primeiro nó passível de ser ligado, nesta situação o nó de nome "node 26".

Para a próxima iteração, tornou-se a aumentar o número de trabalhos em fila de espera, para obrigar o programa a ter de entrar no ciclo de ligar máquinas novamente. O ficheiro de teste da fila de espera usado na 3ª iteração foi o seguinte:

**Ficheiro de teste da fila de espera - 3ª iteração:**

```
eligible jobs----------------------
JOBID           USERNAME      STATE PROCS    WCLIMIT         QUEUETIME

ibm.7289            aba           Idle      14   99:23:59:59  Thu Jan 14 15:51:31
ibm.7289            aba           Idle       6   99:23:59:59  Thu Jan 14 15:51:31

2 eligible jobs
```

Pode-se ver que há agora dois trabalhos em fila de espera, sendo que o primeiro requer 14 processadores. Para o *Cluster* IBM/GridFEUP, se o trabalho em espera requer 14 processadores isso implica que necessita de 7 nós ligados e disponíveis para receber o trabalho em espera. Como já foi observado pela lista de nós, este *Cluster* tem apenas 2 nós *idle* e 2 nós *down*, logo apenas 4 nós disponíveis para receber trabalhos. Portanto, esta alteração permitiu agora testar se ao não haver recursos disponíveis no primeiro *Cluster*, o processo é repetido para o segundo e assim sucessivamente. O resultado desta iteração foi o seguinte:

```
Teste:25 de Junho, 0:41:53
Não há nós Down suficientes para ligar no Cluster GridFEUP/IBM
Teste:25 de Junho, 0:41:53
Os nos a ligar do Cluster Idmec serao:
idmecb17
idmecb13
```

Analisando o resultado obtido é possível observar que o programa reconheceu que o *Cluster* IBM/GridFEUP não tem nós suficientes para receber o trabalho em espera e que efectivamente avançou no processo, testando o segundo *Cluster*. Este, o Idmec, tem 4 processadores por nó, tal como o Ineb. Então, o número de processadores é agora dividido por 4 para verificar quantos nós N deste sistema o trabalho em espera requer. Assim, vem:

Processadores requeridos = 14;

Pela equação (5.1) tem-se: N = (inteiro(14/4)) + 1 <=> N = 3 + 1 <=> N = 4.

O trabalho em espera necessita assim que haja 4 nós disponíveis e ligados no *Cluster* Idmec para poder ser submetido para processamento neste sistema. De referir que agora o critério para somar mais um nó ou não após a divisão do número de processadores por 4 é se esse valor dos processadores é múltiplo de 4. Assim, se for múltiplo de 4 não será somado mais um nó para ser ligado, se não for múltiplo, como o caso exposto no teste, é adicionado mais um nó para ser ligado. Como se pôde observar pela lista de nós, este *Cluster* tem 6 nós desligados e 2 em estado *idle*. Logo, se são necessários 4 para executar o trabalho em espera, é preciso ligar N = 4 - 2 = 2 nós que estão no estado *down*. Verificou-se que o programa respondeu mais uma vez de acordo com o esperado na teoria observando o resultado desta iteração acima exposto.

Também se testou a componente do *script* em que o número de nós ligado é comparado com o valor lido do histórico de cada *Cluster*, com as acções correspondentes consoante o resultado. Para testar essa funcionalidade alterou-se no programa a data de verificação do

histórico. Em vez de estar definida como as 10:00 horas do primeiro dia de cada mês, definiu-se a altura de verificação como sendo no dia 10 de cada mês às 20:01 horas. O resultado do teste foi o seguinte:

**Teste da funcionalidade de verificação do histórico**

**nodesList.out**

Teste:10 de Junho, 20:1:0

compute node summary

*Cluster* IBM/GridFEUP:

Nós *Busy* - Total = 2: node01; node03

Nós *Idle* - Total = 28: node32; node31; node30; node29; node28; node27; node26; node25; node24; node22; node21; node20; node19; node18; node17; node16; node15; node14; node13; node12; node11; node10; node09; node08; node07; node 06; node 05; node 04

Nós *Down* - Total = 1: node23

*Cluster* Idmec:

Nós *Busy* - Total = 36: idmecb19; idmecb18; idmecb17; idmecb16; idmecb15; idmecb14; idmecb13; idmecb12; idmecb11; idmecb09; idmecb08; idmecb06; idmecb05; idmecb04; idmecb03; idmecb02; idmecb01; idmeca20; idmeca18; idmeca17; idmeca16; idmeca15; idmeca14; idmeca13; idmeca12; idmeca11; idmeca10; idmeca09; idmeca08; idmeca07; idmeca06; idmeca05; idmeca04; idmeca03; idmeca02; idmeca01

Nós *Idle* - Total = 0

Nós *Down* - Total = 3: idmecb10; idmecb07; idmeca19

*Cluster* Ineb:

Nós *Busy* - Total = 0

Nós *Idle* - Total = 17: ineb18; ineb17; ineb16; ineb15; ineb14; ineb13; ineb12; ineb11; ineb10; ineb09; ineb08; ineb07; ineb06; ineb05; ineb04; ineb03; ineb02

Nós *Down* - Total = 1: ineb01

```
Teste:10 de Junho, 20:1:0
Os nós a desligar devido ao histórico no Cluster GridFEUP/IBM serão:
node32
node31
node30
node29
node28
node27
node26
node25
node24
node22
node21
node20
node19
node18
node17
node16
node15
node14
node13
node12
node11
Teste:10 de Junho, 20:1:1
Os nós a desligar devido ao histórico no Cluster Ineb serão:
ineb18
ineb17
ineb16
ineb15
ineb14
ineb13
ineb12

Queues.out
Teste:10 de 5, 20:0:59
eligible jobs----------------------
JOBID            USERNAME      STATE PROCS     WCLIMIT          QUEUETIME


0 eligible jobs
```

De acordo com o cálculo do valor médio mensal de utilização de cada *Cluster* exposto no Apêndice B, verifica-se que para o *Cluster* IBM/GridFEUP valor médio de utilização do sistema para o mês de Junho é de 9 nós. Pelos resultados apresentados observa-se que há 30 nós ligados neste *Cluster*, 28 deles em estado *idle*. Desta forma, de acordo com a metodologia

definida, como não há trabalhos em fila de espera só devem permanecer ligados neste *Cluster* o mesmo número de nós do histórico, portanto 9, a não ser que o número de nós em estado *busy* exceda o valor do histórico, sendo que nesse caso se desligariam todos os restantes nós *idle* ligados. No caso apresentado, só existem 2 nós *busy* no *Cluster* IBM/GridFEUP, portanto para manter os 9 nós do histórico apenas era necessário ter mais 7 nós ligados em estado *idle*. Assim, neste caso vem:

- Nós do histórico = 9;
- Nós ligados = 30;
- Nós *idle* = 28;
- Nós *busy* = 2;

O número de nós *idle* N a desligar neste *Cluster* será: N = 28 - (9-2) <=> N = 21.

Logo, deveriam ser desligados 21 nós neste caso devido ao valor do histórico, que como se observa no resultado o *script* retorna a lista de 21 desses 28 nós *idle* para desligar.

Um estudo análogo pode ser feito para os outros *Clusters*. Por conseguinte, para o *Cluster* Idmec vem:

- Nós do histórico = 9;
- Nós ligados = 36;
- Nós *idle* = 0;
- Nós *busy* = 36;

Neste caso, como o número de nós *busy* excede o valor do histórico, deveriam ser desligados todos os nós em estado *idle*. Como não há nenhum nó neste estado neste *Cluster*, nenhum é indicado como passível de ser desligado.

Para o *Cluster* Ineb vem:

- Nós do histórico = 10;
- Nós ligados = 17;
- Nós *idle* = 17;
- Nós *busy* = 0;

O número de nós *idle* N a desligar será: N = 17 - (10-0) <=> N = 7.

Logo, deveriam ser desligados 7 nós neste *Cluster* e como se observa nos resultados uma lista de 7 nós no *Cluster* Ineb é indicada para serem desligados.

Dos resultados obtidos pode-se concluir que o *script* funciona correctamente para toda a gama de acções que eventualmente poderia ter de fazer consoante os diferentes estados do *Cluster* a cada iteração do programa. Não se observaram também quaisquer *crashs* do programa aquando do seu teste durante um período dos ensaios.

## 5.4 – Previsão da Rentabilidade da Solução Idealizada

Nesta secção irá ser feita uma reflexão sobre a rentabilidade da solução encontrada em função dos resultados obtidos. Em teoria, e visto o bom funcionamento do algoritmo demonstrado previamente neste capítulo, este protótipo será rentável e um bom ponto de partida para alcançar uma computação mais eficiente energeticamente e mais "amiga" do ambiente. Esta afirmação pode ser apoiada no facto de não ter qualquer custo físico para a empresa ou entidade que a vai implementar, neste caso o CICA. É um protótipo informático, bastando pôr a executar na estrutura já existente para a solução estar a trabalhar. Este é um

factor importante para a possível obtenção de lucros, ou pelo menos para o possível corte de despesas, numa infra-estrutura como a GridFEUP.

Para esta reflexão sobre a rentabilidade é conveniente definir primeiro algumas variáveis. Para efeitos deste estudo assumir-se-á que os servidores dos *Clusters* da estrutura GridFEUP estão dentro dos parâmetros máximos estabelecidos pela EPA para consumo de potência pelos servidores em estado *idle* apresentados num capítulo anterior deste trabalho. Assim, para o *Cluster* IBM/Grid, sistema em que cada nó possui 2 processadores e uma memória de aproximadamente 4GB, assume-se que tem um gasto energético em estado *idle* de 150W. Relativamente ao *Cluster* Ineb assume-se o mesmo, já que este também tem aproximadamente 4GB de memória e embora apresente 4 *cores*, só dois deles são físicos. O *Cluster* Idmec também tem 4 *cores* mas apresenta cerca de 8GB de memória. Como também estabelecido pela EPA, por cada GB de memória adicional sobre os 4GB típicos, é permitido uma tolerância de 2W por cada GB adicional. No caso do Idmec, como tem 4GB adicionais, assume-se que o seu consumo energético em estado *idle* será de 158W.

Estabelecidos estes parâmetros analisar-se-á a rentabilidade da solução encontrada à luz dos resultados obtidos. Antes ainda, será esclarecido o teor desses resultados com base neles mesmos e numa análise teórica do que aconteceria se fossem efectivamente desligados nós da estrutura GridFEUP. Por conseguinte, a primeira consideração a tomar é que esta estrutura raramente opera a 100%. Durante o período de testes na infra-estrutura GridFEUP houve duas tendências que se revelaram. Uma é que em todos os testes nunca existiu um único trabalho que estivesse em fila de espera para ser submetido. Assim, segundo o algoritmo nunca seria preciso ligar mais máquinas adicionais. Outro facto verificado é que em todos os testes realizados sobre o sistema houve sempre os mesmos 5 nós desligados e todos os restantes estavam ligados, ou em estado *idle* ou em estado *busy*. Outra tendência revelada é que a estrutura GridFEUP tem uma média de utilização de cerca de 31 nós, ou seja, é uma média de cerca de 35% da sua capacidade total. Isto observa-se devido ao facto de em todos os testes o máximo de nós ligados e activos, portanto em estado *busy*, nunca excedeu os 38, sendo nessa altura que se verificou o mínimo de máquinas *idle* encontradas no sistema, 45. Como se percebe este é um número apreciável, sendo que a média aproximada de utilização de nós em estado activo é 31 para os testes efectuados. Ora tendo a estrutura GridFEUP 88 nós no total e não se verificando alteração no número de nós desligados do sistema - 5 - quer isto dizer que genericamente os nós *idle* no sistema são 88 - 31 - 5 = 52. Este é um número bastante elevado. Também se verificou nos testes efectuados que o número de trabalhos activos nunca excedeu os 5 e foi diminuindo mesmo ao longo dos testes. Por aqui se pode inferir que esta estrutura está preparada para ser usada com mais carga de trabalho embora não pareça que a carga de trabalho a que esta estrutura está sujeita seja muito elevada, dando esta boa resposta aos seus utilizadores.

Pelos factores enunciados pode-se retirar a seguinte conclusão: se o mínimo de máquinas *idle* no sistema durante o período dos testes foi de 45 e a sua média é mesmo de 52, com a implementação do protótipo desenvolvido desligar-se-iam, pelo menos, metade destas máquinas a que corresponderia um ganho importante ao nível do consumo de potência e, consequentemente, em termos de custos monetários. Viria então para o mínimo valor de máquinas *idle* que se verificou nos testes e para um consumo médio de energia (CE) na estrutura GridFEUP de (150 + 158 + 150) / 3 =~ 153W/h, sendo 150W/h e 158W/h os valores dos consumos assumidos neste estudo para cada um dos três *Clusters*:

E = Energia consumida em W/h e E = CE * Número de Máquinas *idle* (5.3).

Pela equação (5.3) vem:

E1 = 153 * 45 = 6885W/h = 6.885kW/h.

Se fossem desligadas metade das máquinas como faz o protótipo, viria:

Número de Máquinas *idle* = 45 - (45 / 2) =~ 23.

O gasto energético seria agora:

E2 = 153 * 23 = 3519W/h = 3.519kW/h.

Considerando que um kW/h custa cerca de 0.0742€, a poupança económica P seria:

P = ((E1 –E2) * 0.0742) / 1 (5.4) => P =~ 0.25€.

Ou seja, a poupança por hora seria de 25 cêntimos. Para um dia, a economia seria de P = 0.25 * 24 = 6€. Por mês seria P = 6 * 30 =~ 180€ e por ano andaria à volta de P = 180 * 12 = 2160€.

Assim, haveria uma economia de 3366W/h e de sensivelmente 2200€ por ano se só fosse efectuada uma iteração do protótipo. No entanto, verificou-se pelos resultados obtidos que esse não é o caso. Na iteração seguinte, o mais provável, devido aos factos que se observaram nos testes e que se mencionou anteriormente nesta secção, a tendência seria o *script* implementado continuar a desligar máquinas *idle* o que levaria a reduções dos consumos energéticos e de custos ainda mais elevadas. Imaginando que havia três iterações do algoritmo sem haver mudança na fila de trabalhos, como foi o verificado num dos períodos de testes, é razoável assumir que se conseguiriam desligar, por três vezes, metade dos nós *idle* da infra-estrutura. Consequentemente, os ganhos obtidos seriam ainda maiores pois também mais elevada seria a diferença entre o número de máquinas que estão em média ligadas e o número de servidores que ficariam ligados após estas três iterações do ciclo. Imaginando que uma média de três iterações era feita em que eram desligados nós *idle* sem que depois houvesse grande necessidade de ligar novamente muitos nós adicionais, o que parece ser uma estimativa bastante aceitável tendo em conta o funcionamento verificado na infra-estrutura, por cálculos análogos os ganhos seriam os seguintes:

Ganho energético por hora = 5.967kW/h
Poupança económica diária = 10.63€
Poupança económica mensal =~ 319€
Poupança económica anual =~ 3830€

Efectuou-se ainda outro estudo interessante que foi tentar estimar os ganhos energéticos e monetários do protótipo implementado em relação ao valor médio de utilização da infra-estrutura GridFEUP por mês obtido no histórico calculado no Apêndice B. Como já mencionado, uma estimativa razoável é que três iterações do *script* poderiam ocorrer sem que

a necessidade de ligar nós adicionais fosse muito significativa. Isto pode ser assumido devido ao baixo volume de trabalhos activos na infra-estrutura GridFEUP e a dois factores importantes: O número de nós *down* foi sempre o mesmo, apenas cinco, em todo o período de testes; o número de trabalhos em fila de espera também não se alterou ao longo do mesmo período sendo sempre zero. Outro factor já mencionado foi o facto de que o valor mínimo de máquinas *idle* verificado na GridFEUP em todo o período de ensaios foi 45, o que é um valor considerável. Isto mostra que nunca houve necessidade de ligar nós adicionais e que a estimativa mencionada é bastante aceitável. Devido ao valor de nós *down* ser sempre o mesmo, assumiu-se então para este estudo complementar da rentabilidade da solução que por mês o número máximo de nós *down* foi sempre 10, ou seja, o dobro do verificado no período de testes. Portanto, tendo em conta o valor do histórico que nos indica quantos nós activos houve em cada mês por *Cluster*, pode ser calculado quantos nós activos havia por mês em toda a infra-estrutura, bastando para isso somar os valores da média mensal de utilização de cada *Cluster* em cada mês. Depois, para calcular quantos nós *idle* existiam em média em cada mês é só subtrair ao número de nós total da infra-estrutura GridFEUP o número de nós activos por mês mais os 10 nós *down* anunciados. A tabela 5.1 mostra os resultados obtidos para este cálculo.

Média mensal de nós *down* = 10 nós para todos os meses;

Nós *idle* = Total de nós – (nós *busy* + nós *down*) (5.5)

**Tabela 5.1** – Médias mensais de nós activos e nós em estado *idle* para a infra-estrutura GridFEUP, de acordo com o histórico relativo à utilização de 2009

| **Média Mensal** | Jan | Fev | Mar | Abr | Mai | Jun | Jul | Ago | Set | Out | Nov | Dez |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Nós *Busy* | 34 | 16 | 20 | 27 | 20 | 28 | 23 | 18 | 25 | 20 | 25 | 31 |
| Nós *Idle* | 44 | 62 | 58 | 51 | 58 | 50 | 55 | 60 | 53 | 58 | 53 | 47 |

Para efeitos deste estudo assumiu-se que sempre que a média mensal de nós activos na infra-estrutura GridFEUP for maior ou igual à média aproximada de nós activos verificada no período de testes - 31 – assumiu-se que a média mensal de iterações do algoritmo era uma apenas, ou seja, com o algoritmo implementado seriam desligados por uma vez metade dos nós *idle* da estrutura, sendo assim encontrado o número de nós *idle* com o algoritmo implementado; se a média mensal de nós activos na infra-estrutura GridFEUP for menor ou igual a 20, então assumiu-se que a média mensal de iterações do algoritmo eram três; se a média mensal de nós activos na infra-estrutura GridFEUP estiver entre 20 e 31, então assumiu-se que a média mensal de iterações do algoritmo eram duas.

Portanto, pelo que foi dito anteriormente, observa-se que para o mês de Janeiro e Dezembro se assumiu uma média de apenas uma iteração do algoritmo, para os meses de Fevereiro, Março, Maio, Agosto e Outubro assumiu-se uma média de três iterações do

algoritmo, enquanto para os restantes a média assumida de iterações do algoritmo foi duas. O número de nós *idle* de acordo com o histórico calculado se o algoritmo estivesse implementado na estrutura GridFEUP pode ser consultado na tabela seguinte.

**Tabela 5.2** – Média mensal de nós em estado *idle* para a infra-estrutura GridFEUP, de acordo com o histórico calculado, para o caso do algoritmo estar implementado na infra-estrutura

| **Média Mensal** | Jan | Fev | Mar | Abr | Mai | Jun | Jul | Ago | Set | Out | Nov | Dez |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Nós *Idle* | 22 | 8 | 8 | 13 | 8 | 13 | 14 | 8 | 14 | 8 | 14 | 24 |

Assumindo mais uma vez que o consumo de energia médio (CE) por nó na estrutura GridFEUP é de 153W/h, é só aplicar a equação (5.3) para cada mês e seguir um processo análogo ao que foi feito anteriormente para chegar aos valores dos consumos de energia por mês relativos aos nós em estado *idle* na infra-estrutura GridFEUP. A seguinte tabela mostra esses consumos:

**Tabela 5.3** – Consumos energéticos por mês dos nós em estado *idle* na infra-estrutura GridFEUP sem e com o algoritmo implementado

| **Consumos Energéticos (kW/h)** | Jan | Fev | Mar | Abr | Mai | Jun | Jul | Ago | Set | Out | Nov | Dez |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Sem algoritmo | 6.7 | 9.5 | 8.9 | 7.8 | 8.9 | 7.7 | 8.4 | 9.2 | 8.1 | 8.9 | 8.1 | 7.2 |
| Com algoritmo | 3.4 | 1.2 | 1.2 | 2.0 | 1.2 | 2.0 | 2.1 | 1.2 | 2.1 | 1.2 | 2.1 | 3.7 |

A partir destes valores pode-se obter os custos monetários em cada mês relativos aos nós *idle* na infra-estrutura GridFEUP através das fórmulas seguintes (tendo de novo que um kW/h custa 0.0742€):

Custos por hora = Consumo Energético * 0.0742 (5.6)

Custos por dia = Custos por hora * 24 (5.7)

Custos mensais = Custos por dia * 30 (5.8)

A tabela 5.4 apresentada de seguida exibe os custos económicos referidos.

**Tabela 5.4** – Custos (em €) económicos mensais relativos aos nós em estado *idle* na infra-estrutura GridFEUP sem e com o algoritmo implementado

| **Consumos Energéticos (kW/h)** | Jan | Fev | Mar | Abr | Mai | Jun | Jul | Ago | Set | Out | Nov | Dez |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Sem algoritmo | 360 | 507 | 474 | 417 | 474 | 409 | 450 | 490 | 433 | 474 | 433 | 384 |
| Com algoritmo | 180 | 65 | 65 | 106 | 65 | 106 | 114 | 65 | 114 | 65 | 114 | 196 |

De seguida, apresentam-se duas figuras com informação relevante.

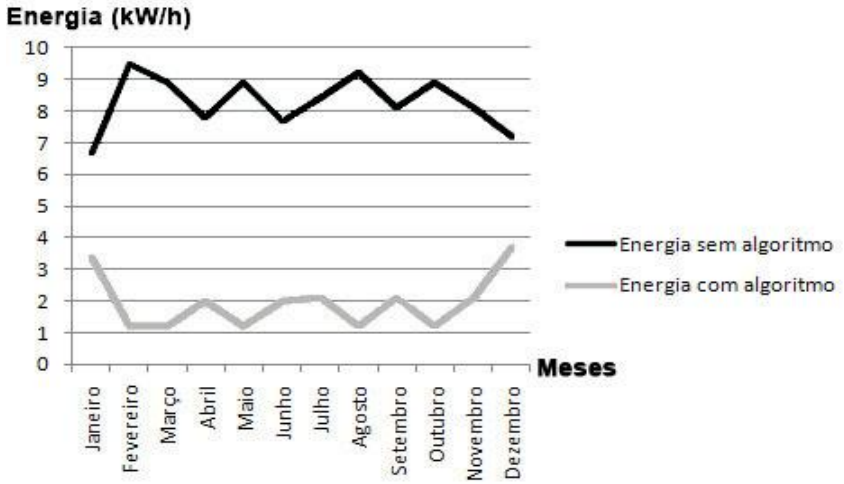



**Figura 5.3** – Gráfico que ilustra a diferença entre os consumos energéticos dos nós em estado *idle* na infra-estrutura GridFEUP com e sem algoritmo implementado

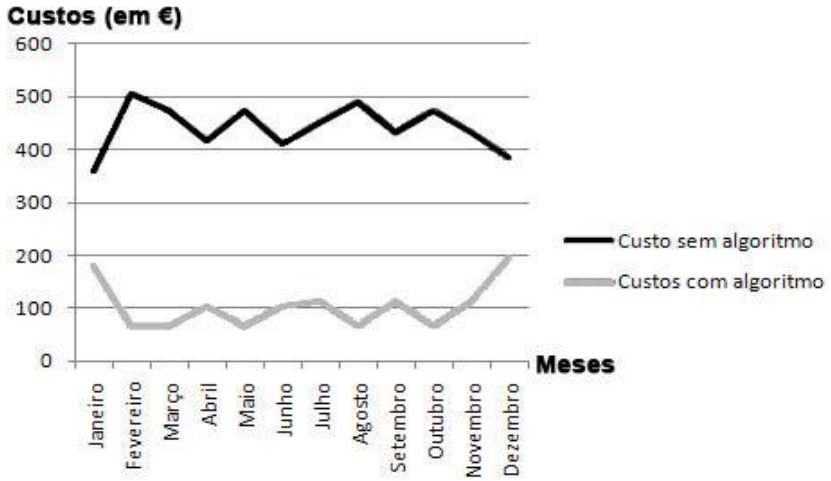



**Figura 5.4** – Gráfico que ilustra a diferença entre os custos económicos dos nós em estado *idle* na infra-estrutura GridFEUP com e sem algoritmo implementado

A figura 5.3 mostra a diferença entre os consumos energéticos dos nós em estado *idle* na infra-estrutura GridFEUP para o caso de se ter o protótipo a funcionar no sistema mencionado ou não. Já a figura 5.4 mostra a diferença entre os custos económicos dos nós em estado *idle* na infra-estrutura GridFEUP para o caso de se ter o protótipo implementado ou não. Os gráficos presentes nestas figuras ajudam a ilustrar melhor os ganhos que se podem obter ao longo do ano devido à solução idealizada. Pode-se concluir das figuras que o algoritmo permite efectivamente obter ganhos em termos energéticos e monetários. Pode ainda observar-se pelos gráficos apresentados nas figuras que as relações entre as linhas que diferenciam os resultados esperados com e sem algoritmo implementado têm uma relação que é aproximadamente inversa, ou seja, quando um cresce o outro diminui. Quer isto dizer que em termos práticos quanto menos a capacidade da infra-estrutura GridFEUP é aproveitada, mais o algoritmo se revela eficaz na economia de energia e custos, já que para meses de menor uso da infra-estrutura os gastos energéticos e os custos aumentam sem algoritmo, enquanto com algoritmo implementado essas parcelas diminuem.

Portanto, pela análise dos resultados e posterior discussão parece que a solução encontrada pode ser fonte de ganhos já consideráveis na infra-estrutura GridFEUP.

A juntar a isto, a infra-estrutura GridFEUP, sendo de elevado desempenho, não é de dimensões muito elevadas comparada com outras existentes. Por exemplo, o Grupo UOL [73] apresentou em Abril deste ano o seu novo *datacenter*, localizado em São Paulo e que tem capacidade para 30 mil servidores. Neste momento, tem 10 mil espalhados por sete locais. Para realizar uma estimativa de ganhos energéticos por desligar metade das máquinas *idle* num sistema deste tipo, poderia ser assumido que a estrutura funciona a 50%, o que não anda muito longe da utilização típica de um *datacenter*. Portanto, dos dez mil servidores cinco mil estariam activos. Dos cinco mil restantes, assume-se que metade está desligada e outra metade em estado *idle*, o que comparando com a estrutura GridFEUP é até uma estimativa optimista. Assim, haveria duas mil e quinhentas máquinas *idle* no *datacenter*. O ganho energético seria agora, assumindo que cada servidor em estado *idle* gasta uma média de 150W:

$$E1 = 150 * 2500 = 375000W/h$$

Se fossem desligadas metade destas máquinas:

$$E2 = 150 * 1250 = 187500W/h$$

$$\text{Ganho Energético por hora} = 375000 – 187500 = 187500W/h$$

Como é lógico, o consumo seria também metade, mas neste caso a diferença era muito considerável. Poupar-se-iam, por hora, 187.5 kW/h de energia, o que é uma economia de energia bastante elevada. Em termos económicos, pela equação (5.4) viria:

Poupança económica diária =~ 334€
Poupança económica mensal = 10020€
Poupança económica anual = 120240€

Verifica-se assim que para um *datacenter* de grandes dimensões os ganhos seriam muito substanciais.

Em jeito de conclusão, o conjunto de circunstâncias verificadas no funcionamento da infra-estrutura GridFEUP leva a crer que numa situação hipotética em que se desligassem praticamente todas as máquinas *idle* do sistema, mesmo que fosse preciso ligar depois algumas máquinas adicionais para responder a cargas de trabalho mais elevadas, o tempo que estas levariam a ligar, tipicamente cerca de dois minutos, e a potência adicional que esses ciclos gastam (cerca de 300 a 400 Watt) não chegaria para retirar os ganhos de eficiência energética que o processo idealizado como solução e implementado no *script* efectuado teria. Esta afirmação também se apoia no facto de não terem existido variações significativas da procura durante o período de observação, havendo quase sempre o mesmo número de trabalhos activos na *Grid*. Esta regularidade permite sustentar, de forma ainda mais intensa, a viabilidade da solução encontrada.

Outra assunção que se poderá fazer é que em termos de temperatura também haveria ganhos em ter menos máquinas ligadas. Dessa forma, haveria menos calor residual dissipado das máquinas e portanto os sistemas de refrigeração beneficiariam desse facto, não precisando de operar com tanta intensidade e gastando menos potência, contribuindo para manter os equipamentos mais saudáveis.

# Capítulo 6

# Conclusão e Trabalho Futuro

## 6.1 – Introdução

Neste capítulo expõem-se então as conclusões retiradas do trabalho e fazem-se algumas considerações sobre trabalhos futuros que podem vir em sequência do trabalho realizado.

## 6.2 – Conclusões

Em jeito de conclusão, o trabalho efectuado permitiu investigar o conceito de *Green Computing* e as tecnologias com ele relacionadas. Assim, observou-se que as infra-estruturas de computação de elevado desempenho como os sistemas de *Cloud Computing, Grid* operacionais ou *Clusters* são infra-estruturas em que o processo de *Green Computing* faz todo o sentido. Isto é devido ao elevado gasto energético de operação destes sistemas e consequente elevado peso económico nos orçamentos das entidades que são proprietárias ou gerem sistemas deste género, geralmente designados de *datacenters*. Esta computação tão poderosa e pouco eficiente em termos energéticos também tem efeitos nocivos para o ambiente.

Pode-se também inferir a partir do trabalho de investigação efectuado que o *Green Computing* começa a ser olhado como forma de maior eficiência energética e poupança económica. No entanto, actualmente este conceito ainda é pouco posto em prática pelos proprietários de *datacenters*, que se preocupam mais com a performance que o sistema tem sem prestar grande atenção à eficiência energética deste.

Conclui-se também que há variados parâmetros já na calha para se tornarem regulamentos ou *standards* pelos quais os *datacenters* e os seus equipamentos se devem reger. Entre esses parâmetros é possível encontrar recomendações sobre temperatura de operação dos *datacenters*, bem como a potência gasta pelos seus servidores, por exemplo, factores que afectam a sua eficiência. Outros parâmetros como o PUE e o DCiE são também importantes neste caso para medir a eficiência energética dos *datacenters*, sendo que começam a ter uma aceitação crescente. Outra conclusão a retirar é que a *standardização* e regulamentação dos *datacenters* tem ainda um longo caminho a percorrer até haver consenso na comunidade das TI sobre esse assunto, havendo mesmo ainda novas métricas a surgir como o CPE e o DCeP.

O projecto envolveu também o desenvolvimento de uma metodologia que permitisse obter ganhos em termos de eficiência energética, desenvolvida a pensar na infra-estrutura GridFEUP. Assim, tomou-se a decisão de criar um *script* que desligasse máquinas na infra-estrutura mencionada que não estivessem a realizar "trabalho útil". Para isso, o algoritmo pensado está ligado ao escalonador do sistema e, consoante o estado dos nós da GridFEUP, toma as decisões mais convenientes de quando e quais as máquinas a ligar ou desligar para se obter uma boa performance tanto a nível de eficiência energética como a nível da execução dos trabalhos no sistema.

Também no âmbito deste projecto implementou-se um protótipo sobre a estrutura GridFEUP que se relaciona com o seu escalonador e tem por base a metodologia idealizada anteriormente. Após os testes realizados no sistema verificou-se que o protótipo apresenta um funcionamento correcto e os resultados obtidos foram bastante satisfatórios, permitindo concluir que o programa implementado parece ser realmente uma boa solução. O protótipo nunca apresentou quebras no seu funcionamento e os resultados obtidos corresponderam ao que teoricamente era esperado alcançar, o que atesta da viabilidade da solução.

Finalmente, também se conclui que a solução implementada é aparentemente rentável, tanto mais porque não acarreta qualquer custo e, portanto, os ganhos apresentados em termos energéticos e em termos económicos significam um "lucro" de 100% em relação ao que se verificava anteriormente no sistema. Outra conclusão possível é que indirectamente a solução obtida ajudará ainda a manter os equipamentos mais saudáveis, já que gastam menos potência e produzem menos energia dissipada, trabalhando a menores temperaturas e estando ligados por períodos bastante menores.

## 6.3 – Trabalho Futuro

Este projecto tem bastante potencial para ser continuado no futuro. O protótipo implementado pode ser ainda mais optimizado para que os ganhos que advêm da sua implementação sejam ainda maiores. Dito isto, há algumas medidas que podem ajudar a uma melhor optimização do algoritmo, como as que se apresentam de seguida.

Assim, a mais óbvia continuação deste trabalho será implementar no algoritmo a funcionalidade de verificar qual o melhor nó a desligar consoante o nível que o nó exibe de parâmetros como temperatura e/ou potência. Para isso acontecer, teria de se arranjar uma forma de ler esses parâmetros nos nós dos *Clusters*, algo que ainda não é possível com o *software* actual. Claro que outros parâmetros energéticos relevantes poderiam também ser incluídos.

Outro avanço da solução implementada tem a ver com a questão do histórico de utilização dos *Clusters*. Seria interessante encontrar uma forma de obter automaticamente as estatísticas diárias de utilização de cada *Cluster*, sem ser assim necessária grande computação por parte dos computdores. Com uma estatística diária, o algoritmo poderia verificar diariamente qual o valor médio de nós que o *Cluster* teve ligado num determinado dia e, além disso, ao ser feito de forma automática no *script*, este valor do histórico iria ser calculado recorrendo efectivamente a nós do sistema e não a uma aproximação como o histórico efectuado para efeitos de teste do protótipo implementado.

Outro melhoramento do protótipo poderá partir da previsão de quanto tempo demorarão os trabalhos a ser executados e qual a velocidade de execução das máquinas. Esta funcionalidade permitiria assim tomar uma decisão mais informada sobre o melhor nó a escolher para cada trabalho, relacionando a velocidade de execução de cada máquina com o tempo previsto que cada trabalho demorará. O escalonador Moab apresenta sempre uma estimativa de duração restante dos trabalhos mas, no entanto, verificou-se nos ensaios realizados que essa estimativa não era muito exacta. Portanto, também o escalonador teria de ter informações mais exactas nesse aspecto. A velocidade de execução de cada máquina poderia partir de um critério de carga e processadores já em uso de cada máquina, sendo que uma máquina menos carregada executará mais rápido que uma mais carregada.

# Referências

[1] A. Iosup, C. Dumitrescu, D. Epema, Hui Li, and L. Wolters, “How are Real Grids Used? The Analysis of Four Grid Traces and Its Implications”, in *7th IEEE/ACM International Conference on Grid Computing*, September 2006.

[2] A. -C. Orgerie, "Energy-Aware for High-Performance Data Transport and Computations in Large-Scale Distributed Systems", Rapport de Stage, Master d'Informatique Fondamentale, ENS Lyon.

[3] http://www.fe.up.pt/si/web_base.gera_pagina?P_pagina=21101, acedido pela última vez em 27-06-2010.

[4] L. Lefèvre and J. -M. Pierson, "Energy Savings in ICT and ICT for Energy Savings", in *ERCIM NEWS Number 79, October 2009: Towards Green ICT*, pp. 10-11.

[5] http://www.green500.org, acedido pela última vez em 27-06-2010.

[6] http://www.thegreengrid.org, acedido pela última vez em 27-06-2010.

[7] www.cost804.org, acedido pela última vez em 27-06-2010.

[8] A. Phippen, "Who Makes IT Professionals Environmentally Aware?", in *ERCIM NEWS Number 79, October 2009: Towards Green ICT*, p. 16.

[9] H. Engelstaedter, "Combine Cost Optimization and Energy Efficiency Targets in your IT Strategy: The Going Green Impact Tool Supports You", in *ERCIM NEWS Number 79, October 2009: Towards Green ICT*, pp. 17-18.

[10] U. Barth, P. Wong and D. Bourse, "Key Challenges for Green Networking", in *ERCIM NEWS Number 79, October 2009: Towards Green ICT*, pp. 13-14.

[11] C. Öhman, "Design for Energy Awareness", in *ERCIM NEWS Number 79, October 2009: Towards Green ICT*, p. 15.

[12] K. Winkler, V. Avelar, W. Torell and T. Hampel, "Data Center Baseline Study Report", April 2008.

[13] A. -C. Orgerie, "Save Watts in your Grid: Green Strategies for Energy-Aware Framework in Large Scale Distributed Systems", in *14th IEEE International Conference on Parallel and Distributed Systems*, 2008.

[14] A.-C. Orgerie, L. Lefèvre and J.-P. Gelas, "Chasing gaps between bursts : Towards energy efficient large scale experimental grids", in *PDCAT 2008 : The Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies, Dunedin, New Zealand*, December 2008.

[15] F. Hermenier, N. Loriant and J. -M. Menaud, "Power Management in Grid Computing with Xen".

[16] www.patentstorm.us/patents/6728800/description.html, acedido pela última vez em 27-06-2010.

[17] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt and A. Warfield, "Live Migration of Virtaula Machines", in *Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation (NSDI '05)*, Boston, MA, USA, 2005.

[18] V. Keller and W. Ziegler, "ÏANOS: How to Do More Science with Less Energy", in *ERCIM NEWS Number 79, October 2009: Towards Green ICT*, pp. 26-27.

[19] G. I. Meijer, T. Brunschwiler, S. Paredes and B. Michel, "Using Waste Heat from Data Centres to Minimize Carbon Dioxide Emission", in *ERCIM NEWS Number 79, October 2009: Towards Green ICT*, pp. 23-24.

[20] T. Hey and A. E. Trefethen, "The UK e-Science Core Programme and the Grid", in *Journal of Future Generation Computer Systems (FGCS)*, vol. 18, no. 8, pp. 1017-1031, 2002.

[21] I. Foster and C. Kesselman, "The Grid. Blueprint for a new Computing Infrastructure", Morgan Kauffman, 1998.

[22] H. Stockinger, "Defining the grid: a snapshot on the current view", 2007.

[23] I. Foster and S. Tuecke, "Describing the elephant: the different faces of IT as services", ACM Queue 3(6) : pp. 26-29, 2005.

[24] R. Buyya and S. Venugopal, "A Gentle Introduction to Grid Computing and Technologies", July 2005.

[25] www.ogf.org, acedido pela última vez em 27-06-2010.

[26] I. Foster, C. Kesselman and S. Tuecke, "The anatomy of the grid: enabling scalable virtual organizations" Int J Supercomput Appl 15(3): pp. 200–222, 2001.

[27] W. Kim, "Cloud Computing: Today and Tomorrow", in *Journal of Object Technology*, vol. 8, no. 1, 2009.

[28] http://en.wikipedia.org/wiki/Cloud_computing#Architecture, acedido pela última vez em 27-06-2010.

[29] http://searchcloudcomputing.techtarget.com/sDefinition/0,,sid201_gci1287881,00.html, acedido pela última vez em 27-06-2010.

[30] www.infoworld.com/d/cloud-computing/what-cloud-computing-really-means-031, acedido pela última vez em 27-06-2010.

[31] http://developer.amazonwebservices.com/connect/entry.jspa?externalID=1632& categoryID=100, acedido pela última vez em 27-06-2010.

[32] C. Belady, A. Rawson, J. Pfleuger and T. Cader, "Green Grid Data Center Power Efficiency Metrics: PUE and DCIE", 2008.

[33] A. Fanara, E. Haines and A. Howard, "The State of Energy and Performance Benchmarking for Enterprise Servers". Available at www.springerlink.com.

[34] http://pt.wikipedia.org/wiki/Clusters, acedido pela última vez em 27-06-2010.

[35] www.beowulf.org/, acedido pela última vez em 27-06-2010.

[36] M. Baker, R. Buyya and D. Hyde, "Cluster Computing: A High-Performance Contender", in *Technical Activities Forum*, July 1999.

[37] http://en.wikipedia.org/wiki/Cluster_(computing), acedido pela última vez em 27-06-2010.

[38] M. Baker and R. Buyya, "Cluster Computing: The Commodity Supercomputer", 1999.

[39] www.top500.org, acedido pela última vez em 27-06-2010.

[40] www.globus.org, acedido pela última vez em 27-06-2010.

[41] www.anl.gov, acedido pela última vez em 27-06-2010.

[42] http://public.web.cern.ch/public/, acedido pela última vez em 27-06-2010.

[43] www.clusterresources.com/products/mwm/docs/moabusers.shtml, acedido pela última vez em 27-06-2010.

[44] www.clusterresources.com/products/mwm/docs/index.shtml, acedido pela última vez em 27-06-2010.

[45] www.hpcprojects.com/products/product_details.php?product_id=255, acedido pela última vez em 27-06-2010.

[46] http://ganglia.info/?p=45, acedido pela última vez em 27-06-2010.

[47] www.ibm.com/developerworks/linux/library/l-ganglia-nagios-1/index.html?S_TACT=105AGY79&S_CMP=content, acedido pela última vez em 27-06-2010.

[48] J. Loper and S. Parr, “Energy Efficiency in Data Centers: A New Policy Frontier”, January 2007.

[49] http://www.epa.gov/, acedido pela última vez em 27-06-2010.

[50] R. Sawyer, “Calculating Total Power Requirements for Data Centers”, White Paper #3, 2004, American Power Conversion.

[51] http://www.ashrae.org, acedido pela última vez em 27-06-2010.

[52] http://www.greenit-pc.jp/e/index.html, acedido pela última vez em 27-06-2010.

[53] http://www.energystar.gov/index.cfm?c=about.ab_index, acedido pela última vez em 27-06-2010.

[54] “Energy Star Program Requirements for Computers, Version 5.0”, November 2008.

[55] “Energy Star Program Requirements for Computer Servers, Final Draft”, April 2009.

[56] http://arstechnica.com/civis/viewtopic.php?f=8&t=354552, acedido pela última vez em 27-06-2010.

[57] http://pt.wikipedia.org/wiki/Framebuffer, acedido pela última vez em 27-06-2010.

[58] M. K. Patterson, “The Effect of Data Center Temperature on Energy Efficiency”.

[59] ASHRAE, “Thermal Guidelines for Data Processing Environments”, 1st Edition, ASHRAE, Atlanta, 2004.

[60] F. Fallah and M. Pedram, “Standby and Active Leakage Current Control and Minimization in CMOS VLSI Circuits”, in *IEICE Transactions on Electronics*, 2005.

[61] R. Mukherjee et. al., “Peak Temperature Control and Leakage Reduction During Binding in High Level Synthesis”, August 2005.

[62] F. S. Dubin and C. Long, *Energy Conservation Standards for Building Design Construction and Operation* 1st Edition, McGraw Hill, 1978.

[63] A. Rawson, J. Pfleuger and T. Cader, “Green Grid Data Center Power Efficiency Mterics: PUE and DCiE”, White Paper #6, 2008.

[64] C. Belady, “How to Minimize Data Center Utility Bills”, September 2006.

[65] http://www.42u.com/measurement/pue-dcie.htm, acedido pela última vez em 27-06-2010.

[66] http://www.42u.com/measurement/dcep.htm, acedido pela última vez em 27-06-2010.

[67] http://www.datacenterdynamics.com/ME2/Audiences/dirmod.asp?sid=4015BD1DA6DE4E5A9CFBFE0FD8A9D12C&nm=Magazine+Archives&type=Publishing&mod=Publications%3A%AArticle&mid=8F3A7027421841978F18BE895F87F791&tier=4&id=AA31C266434D4801B751E418AAE26DFA&AudID=E5BD2FF22AF74DF3A0D5F4E519A61511, acedido pela última vez em 27-06-2010.

[68] C. Belady and C. Malone, “A Metric and Infrastructure Model to Evaluate Data Center Efficiency”, 2007.

[69] http://support.rightscale.com/06-FAQs/FAQ_0028_-_How_do_the_CPU _Metrics_work _and_what_is_CPU_Steal%3f, acedido pela última vez em 27-06-2010.

[70] http://www.gridlab.org/WorkPackages/wp-11/monitor-manual/metrics.html, acedido pela última vez em 27-06-2010.

[71] http://docs.hp.com/en/B2355-90672/ch06s02.html, acedido pela última vez em 27-06-2010.

{72] http://www.clusterresources.com/products/mwm/index.shtml, acedido pela última vez em 27-06-2010.

[73] J. A. S. Augusto, “Manual Conciso de Perl”, Outubro de 2001.

[74] http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.csm16010.cmds.doc/am7cm_rpower.html, acedido pela última vez em 27-06-2010.

[75] http://ipmitool.sourceforge.net/manpage.html, acedido pela última vez em 27-06-2010.

[76] http://tecnologia.uol.com.br/ultimas-noticias/redacao/2010/04/27/com-capacidade-para-30-mil-servidores-novo-data-center-do-uol-coloca-computacao-verde-em-pratica.jhtm,acedido pela última vez em 27-06-2010.

[77] J. C. Ferreira, J. C. Lopes e J. M. da Silva, “Norma de Formatação e Orientações para a Escrita de Dissertações ou Relatórios de Projecto do MIEEC, Maio de 2008.

# Apêndice A

A matriz apresentada de seguida apresenta os vários parâmetros e métricas encontrados e explicitados no capítulo 5 deste trabalho. Os valores por referência, quando aplicável, são também mostrados na tabela.

**Matriz dos *Standards* e Parâmetros energéticos**

| **Parâmetros e *Standards* Energéticos** | **Valores de referência** |
|---|---|
| **Potência – Especificação da EPA para computadores** | TEC (kWh)<br><br>Categoria A: <= 148.0<br>Categoria B: <= 175.0<br>Categoria C: <= 209.0<br>Categoria D: <= 234.0 |
| **Potência – Especificação da EPA para servidores** | Limite de potência em estado *idle* (W)<br><br>Categoria A: 55.0<br>Categoria B: 65.0<br>Categoria C: 100.0<br>Categoria D: 150.0 |
| **Temperatura – segundo recomendações da ASHRAE** | Gama aceitável: 15ºC – 32ºC<br>Gama recomendável: 20ºC – 25ºC |

| | |
|---|---|
| **PUE** | 1.2 ⇔ Muito Eficiente<br>1.5 ⇔ Eficiente<br>2.0 ⇔ Eficiência Razoável / Média<br>2.5 ⇔ Ineficiente<br>3.0 ⇔ Muito Ineficiente |
| **DCiE** | 83% ⇔ Muito Eficiente<br>67% ⇔ Eficiente<br>50% ⇔ Eficiência Razoável / Média<br>40% ⇔ Ineficiente<br>33% ⇔ Muito Ineficiente |
| **DCeP** | Não aplicável |
| **CPE** | Não aplicável |
| **Velocidade das Ventoinhas dos Servidores** | Não aplicável |
| **Pacotes Enviados / Recebidos** | Não aplicável |
| **Espaço Total no Disco / Espaço Disponível no Disco** | Não aplicável |
| **Contagem de CPUs** | Não aplicável |
| **Velocidade do CPU** | Não aplicável |
| **CPU User** | Não aplicável |
| **CPU Nice** | Não aplicável |
| **CPU System** | Não aplicável |
| **CPU WaitIO** | Não aplicável |
| **CPU Idle** | Não aplicável |
| **Média da carga de trabalho do servidor** | Não aplicável |

| | |
|---|---|
| **Swap Livre** | Não aplicável |
| **Memória Total** | Não aplicável |
| **Memória Livre** | Não aplicável |
| **Total de Processos** | Não aplicável |
| **Total de Processos a Correr** | Não aplicável |
| **Bytes In / Out** | Não aplicável |

# Apêndice B

O histórico da média mensal de nós utilizados em cada *Cluster* foi calculado com base nas estatísticas relativas à utilização da estrutura GridFEUP no ano de 2009. Estas são exibidas de seguida:

*Cluster* IBM/GridFEUP:

| GRUPO | Jan | Fev | Mar | Abr | Mai | Jun | Jul | Ago | Set | Out | Nov | Dez |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CEHRA | 11 | 8 | 11 | 7 | 7 | 0,5 | 2 | 10 | 53 | 19 | | 13 |
| CESA | 22 | 60 | 61 | 56 | 77 | 87 | 13 | 28 | | 8 | 11 | |
| CICA | | | 0 | 0 | 0 | | | | | | | |
| DEEC | | | 11 | 20 | | 0 | 4 | 3 | 0 | 1 | | |
| GERAL | 28 | 0 | | | | 0,5 | 33 | 19 | 1 | 0 | 21 | 33 |
| LSRE | 39 | 32 | 17 | 17 | 16 | 12 | 48 | 40 | 46 | 72 | 68 | 54 |

*Cluster* Idmec:

| GRUPO | Jan | Fev | Mar | Abr | Mai | Jun | Jul | Ago | Set | Out | Nov | Dez |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| x | | | | | 16 | 21 | 29 | 7 | 9 | 26 | | |
| X | | 0 | 0 | | | 0 | 0,5 | | | | | |
| X | 28,5 | 88 | 85 | 86 | 63 | 17 | 5 | | 43,5 | 8 | 0 | |
| X | | 1 | 12 | 12 | 10 | 3 | 29 | 9 | 4 | 2 | 37 | 100 |
| X | 0 | | | | | | | | | | 26 | |
| X | | | | | | 0 | | | 0 | | | |
| X | 71,5 | 2 | 2 | 2 | 3 | 1 | | | | | | |
| X | | | 0 | 0 | 0 | | | | 0 | 0 | 37 | |
| X | | | | 0 | | | 0,5 | 1 | 0 | 0 | | |
| x | 0 | 9 | 1 | | 8 | 58 | 36 | 83 | 43,5 | 64 | | |

*Cluster* Ineb:

| GRUPO | Jan | Fev | Mar | Abr | Mai | Jun | Jul | Ago | Set | Out | Nov | Dez |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X | 7 | | | | | | | | | | | |
| X | 7 | | | | | | | | | | | |
| X | 48 | | | | | | | | | | | |
| X | | | | | | 24 | 100 | | | | | |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X | 24 | | | | | | | | | | | |
| X | | | | | | | | | 93 | | | |
| X | 14 | 100 | | 100 | | 76 | 0 | | 7 | | 100 | |

As estatísticas utilizadas são relativas à percentagem de uso de cada grupo que submeteu trabalhos na infra-estrutura GridFEUP em cada mês de 2009. Este uso está nesta estatística medido em 'Horas de utilização dos processadores'. No entanto, isto não nos permitia chegar ao número de nós usado efectivamente por cada grupo. Assim, e após consulta de membros gestores da estrutura, considerou-se que era uma aproximação viável e aceitável considerar esta estatística como percentagem de processadores utilizados em cada *Cluster*, até porque este histórico foi realizado para efeitos de teste do protótipo implementado.

O processo de cálculo da média mensal de utilização de nós é apresentado de seguida.

Primeiro, calcula-se quantos processadores tem cada *Cluster*. Vem então:

IBM/GridFEUP: 31 nós e 2 processadores por nó => Pgr = 31 * 2 = 62 proc;

Idmec: 39 nós e 4 processadores por nó => Pid = 39 * 4 = 156 proc;

Ineb: 18 nós e 4 processadores por nó => Pin = 18 * 4 = 72 proc;

onde Px = número de processadores de cada *Cluster*.

Depois, calcula-se quantos nós usa cada grupo em cada mês. Para isso, temos:

Per = Percentagem de utilização de nós do grupo 'x';

PNx = Processadores por nó em cada *Cluster*;

NPx = Número de processadores usado pelo grupo 'x'.

Primeiramente calcula-se quantos processadores usa cada grupo pela seguinte fórmula:

$$NPx = (Px * Per) / 100 \quad (1)$$

Posto isto, o número de nós Nx que o grupo 'x' utilizou em cada mês obtém-se por:

$$Nx = NPx / PNx \quad (2)$$

Tendo o número de nós por grupo em cada mês calculado, basta agora calcular a média mensal M que será então o valor do histórico de ocupação do *Cluster* num dado mês. Vem assim:

$$M = \Sigma Nx / G \quad (3),$$

sendo G = número de grupos que submeteram trabalho no *Cluster* correspondente num determinado mês.

Para exemplificar o processo, calcularemos agora a média de ocupação do *Cluster* para o mês de Janeiro no sistema IBM/GridFEUP. Temos que:

G = 4; Px = 62;

Grupo CEHRA: Por (1), vem: NPcehra = (62 * 11) / 100 NPcehra = 6.82 => NPcehra = 7 proc
Por (2), vem: Ncehra = 7 / 2 = 3.5 ~= 4 nós

Grupo CESA: Por (1), vem: NPcesa = (62 * 22) / 100 = 13.64 => NPcesa = 14 proc
Por (2), vem: Ncesa = 14 / 2 = 7 nós

Grupo GERAL: Por (1), vem: NPgeral = (62 * 28) / 100 = 17.36 => NPgeral = 18 Proc

Por (2), vem:

Ngeral = 18 / 2 = 9 nós

Neste caso, aproximou-se 17.36 para 18 e não 17 pois o resultado indica que mais do que 17 processadores foram usados, logo têm de ser 18. Este processo de aproximação foi usado em todos os cálculos, tanto para número de processadores como para número de nós.

Grupo LSRE: Por (1), vem: NPlsre = (62 * 11) / 100 = 24.18 => NPlsre = 25 proc
Por (2), vem: Ncehra = 25 / 2 □ Ncehra = 12.5 ~= 13 nós

Finalmente, por (3) obtém-se: M = (4 + 7 + 9 + 13) / 4 = 8.25 => M = 9 nós

Os históricos obtidos são assim:

*Cluster* IBM/GridFEUP:

| **GRUPO** | **Jan** | **Fev** | **Mar** | **Abr** | **Mai** | **Jun** | **Jul** | **Ago** | **Set** | **Out** | **Nov** | **Dez** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CEHRA | 4 | 3 | 4 | 3 | 3 | 1 | 1 | 4 | 17 | 6 | | 5 |
| CESA | 7 | 19 | 19 | 18 | 24 | 27 | 5 | 9 | | 3 | 4 | |
| CICA | | | | | | | | | | | | |
| DEEC | | | 4 | 7 | | | 2 | 1 | | 1 | | |
| GERAL | 9 | | | | | 1 | 11 | 6 | 1 | | 7 | 11 |
| LSRE | 13 | 10 | 6 | 6 | 5 | 4 | 15 | 13 | 15 | 23 | 22 | 17 |
| | | | | | | | | | | | | |
| Média Mensal | 9 | 11 | 9 | 9 | 11 | 9 | 7 | 7 | 11 | 9 | 11 | 11 |

*Cluster* Idmec:

| GRUPO | Jan | Fev | Mar | Abr | Mai | Jun | Jul | Ago | Set | Out | Nov | Dez |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X | | | | | 7 | 9 | 12 | 3 | 4 | 11 | | |
| X | | | | | | | 1 | | | | | |
| X | 12 | 35 | 34 | 34 | 25 | 7 | 2 | | 17 | 4 | | |
| X | | 1 | 5 | 5 | 4 | 2 | 12 | 4 | 2 | 1 | 11 | 20 |
| X | | | | | | | | | | | 8 | |
| X | | | | | | | | | | | | |
| X | 28 | 1 | 1 | 1 | 2 | 1 | | | | | | |
| X | | | | | | | | | | | 11 | |
| X | | | | | | | 1 | 1 | | | | |
| X | | 4 | 1 | | 4 | 23 | 15 | 33 | 17 | 25 | | |
| Média Mensal | 20 | 11 | 11 | 14 | 9 | 9 | 8 | 11 | 10 | 11 | 10 | 20 |

De referir que para este *Cluster*, após estudo do número de trabalhos processados pelo sistema em cada mês, decidiu-se assumir que em Novembro a utilização do sistema foi de 70% e 50% para Dezembro, o que significa que para efeitos do cálculo do histórico o número total de processadores baixou de 156 para 110 no caso de Novembro e 78 em Dezembro.

*Cluster* Ineb:

| GRUPO | Jan | Fev | Mar | Abr | Mai | Jun | Jul | Ago | Set | Out | Nov | Dez |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X | 2 | | | | | | | | | | | |
| X | 2 | | | | | | | | | | | |
| X | 9 | | | | | | | | | | | |
| X | | | | | | 5 | 8 | | | | | |
| X | 5 | | | | | | | | | | | |
| X | | | | | | | | | 7 | | | |
| X | 3 | 4 | | 4 | | 14 | | | 1 | | 4 | |
| | | | | | | | | | | | | |
| Média Mensal | 5 | 4 | 0 | 4 | 0 | 10 | 8 | 0 | 4 | 0 | 4 | 0 |

Também para o *Cluster* Ineb se assumiu diferentes utilizações em diferentes meses conforme o número de trabalhos submetidos em cada mês. Assim, para os meses de Fevereiro, Abril e Novembro assumiu-se uma utilização de apenas 20% da infra-estrutura, o que resulta que o número de processadores baixou de 72 para 15 nos cálculos realizados para estes meses. Já para os meses de Julho e Setembro assumiu-se uma utilização de 40%, ou seja, o número de processadores para efeitos de cálculo foi de 29.

# Apêndice C

Aqui apresenta-se o formato dos resultados que o protótipo apresenta:

**1**

**Primeiro Momento**

**nodesList.out**

Teste:10 de Junho, 10:3:27

compute node summary

| Name | State | Procs | Memory | Opsys |
|---|---|---|---|---|
| node32 | Idle | 2:2 | 3945:3945 | linux |
| node31 | Idle | 2:2 | 3948:3948 | linux |
| node30 | Idle | 2:2 | 3945:3945 | linux |
| node29 | Idle | 2:2 | 3945:3945 | linux |
| node28 | Idle | 2:2 | 3945:3945 | linux |
| node27 | Idle | 2:2 | 3945:3945 | linux |
| node26 | Idle | 2:2 | 3945:3945 | linux |
| node25 | Idle | 2:2 | 3945:3945 | linux |
| node24 | Idle | 2:2 | 3945:3945 | linux |
| node23 | Down | 0:2 | 1:1 | DEFAULT |
| node22 | Idle | 2:2 | 3945:3945 | linux |
| node21 | Idle | 2:2 | 3945:3945 | linux |
| node20 | Idle | 2:2 | 3945:3945 | linux |
| node19 | Idle | 2:2 | 3948:3948 | linux |
| node18 | Idle | 2:2 | 3945:3945 | linux |
| node17 | Idle | 2:2 | 3945:3945 | linux |
| node16 | Idle | 2:2 | 3948:3948 | linux |

```
node15          Idle   2:2    3945:3945    linux
node14          Idle   2:2    3945:3945    linux
node13          Idle   2:2    3945:3945    linux
node12          Idle   2:2    3945:3945    linux
node11          Idle   2:2    3945:3945    linux
node10          Idle   2:2    3945:3945    linux
node09          Idle   2:2    3945:3945    linux
node08          Idle   2:2    3945:3945    linux
node07          Idle   2:2    3945:3945    linux
node06          Idle   2:2    3945:3945    linux
node05          Idle   2:2    3945:3945    linux
node04          Idle   2:2    3945:3945    linux
node03          Busy   0:2    3945:3945    linux
node01          Busy   0:2    3945:3945    linux
idmecb19        Busy   0:4    7487:7487    linux
idmecb18        Busy   0:4    7487:7487    linux
idmecb17        Busy   0:4    7487:7487    linux
idmecb16        Busy   0:4    7487:7487    linux
idmecb15        Busy   0:4    7487:7487    linux
idmecb14        Busy   0:4    7487:7487    linux
idmecb13        Busy   0:4    7487:7487    linux
idmecb12        Busy   0:4    7487:7487    linux
idmecb11        Busy   0:4    7487:7487    linux
idmecb10        Down   0:4    5467:5467    linux
idmecb09        Busy   0:4    7487:7487    linux
idmecb08        Busy   0:4    7487:7487    linux
idmecb07        Down   0:4    7487:7487    linux
idmecb06        Busy   0:4    7487:7487    linux
idmecb05        Idle   4:4    7487:7487    linux
idmecb04        Idle   4:4    7487:7487    linux
idmecb03        Busy   0:4    7487:7487    linux
idmecb02        Busy   0:4    7487:7487    linux
idmecb01        Busy   0:4    7487:7487    linux
idmeca20        Idle   4:4    7487:7487    linux
idmeca19        Down   0:4    1:1          DEFAULT
idmeca18        Idle   4:4    7487:7487    linux
idmeca17        Idle   4:4    7487:7487    linux
idmeca16        Idle   4:4    7487:7487    linux
```

```
idmeca15            Idle   4:4     7487:7487      linux
idmeca14            Idle   4:4     7487:7487      linux
idmeca13            Idle   4:4     7487:7487      linux
idmeca12            Idle   4:4     7487:7487      linux
idmeca11            Idle   4:4     7487:7487      linux
idmeca10            Idle   4:4     7487:7487      linux
idmeca09            Busy   0:4     7487:7487      linux
idmeca08            Busy   0:4     7487:7487      linux
idmeca07            Busy   0:4     7487:7487      linux
idmeca06            Busy   0:4     7487:7487      linux
idmeca05            Busy   0:4     7487:7487      linux
idmeca04            Busy   0:4     7487:7487      linux
idmeca03            Busy   0:4     7487:7487      linux
idmeca02            Busy   0:4     7487:7487      linux
idmeca01            Busy   0:4     7487:7487      linux
ineb18              Idle   4:4     3879:3879      linux
ineb17              Idle   4:4     3879:3879      linux
ineb16              Idle   4:4     3879:3879      linux
ineb15              Idle   4:4     3879:3879      linux
ineb14              Idle   4:4     3879:3879      linux
ineb13              Idle   4:4     3879:3879      linux
ineb12              Idle   4:4     3879:3879      linux
ineb11              Idle   4:4     3879:3879      linux
ineb10              Idle   4:4     3879:3879      linux
ineb09              Idle   4:4     3879:3879      linux
ineb08              Idle   4:4     3879:3879      linux
ineb07              Idle   4:4     3879:3879      linux
ineb06              Idle   4:4     3879:3879      linux
ineb05              Idle   4:4     3879:3879      linux
ineb04              Idle   4:4     3879:3879      linux
ineb03              Idle   4:4     3879:3879      linux
ineb02              Idle   4:4     3879:3879      linux
ineb01              Down   0:4       1:1       DEFAULT
-----               ---  172:290  466791:466791    -----

Total Nodes: 88  (Active: 26  Idle: 57  Down: 5)

Teste:10 de Junho, 10:3:27
```

Os nos a desligar do Cluster GridFEUP/IBM serao:
node32
node31
node30
node29
node28
node27
node26
node25
node24
node22
node21
node20
node19
node18
Teste:10 de Junho, 10:3:27
Os nos a desligar do Cluster Idmec serao:
idmecb05
idmecb04
idmeca20
idmeca18
idmeca17
idmeca16
Teste:10 de Junho, 10:3:27
Os nos a desligar do Cluster Ineb serao:
ineb18
ineb17
ineb16
ineb15
ineb14
ineb13
ineb12
ineb11

**Queues.out**

Teste:10 de 5, 10:3:27

active jobs----------

```
JOBID           USERNAME     STATE PROCS   REMAINING          STARTTIME

ibm.8406        dce05015    Running    1 97:05:46:08  Mon Jun  7 15:49:36
ibm.8407        dce05015    Running    1 97:05:47:46  Mon Jun  7 15:51:14
ibm.8408        dce05015    Running    1 97:05:49:14  Mon Jun  7 15:52:42
ibm.8409        dce05015    Running    1 97:05:50:40  Mon Jun  7 15:54:08

4 active jobs           4 of 272 processors in use by local jobs (1.47%)
                     2 of 83 nodes active      (2.41%)

eligible jobs----------------------
JOBID           USERNAME     STATE PROCS     WCLIMIT          QUEUETIME



0 eligible jobs

blocked jobs-----------------------
JOBID           USERNAME     STATE PROCS     WCLIMIT          QUEUETIME

ibm.7289             aba  BatchHold     4 99:23:59:59  Thu Jan 14 15:51:31
ibm.7296             aba  BatchHold     4 99:23:59:59  Thu Jan 14 16:41:02

2 blocked jobs

Total jobs:  6
```